Approved for public release; distribution is unlimited

DEVELOPMENT OF DYNAMIC, CONTEXT-DRIVEN HELP SYSTEMS

by

Nancy Jo McClellan
Lieutenant, United States Navy
B. S. Business Administration, University of Kansas, 1984

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL

March 1993

REPORT DOCUMENTATION PAGE

REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution is unlimited
DECLASSIFICATION/DOWNGRADING SCHEDULE	
ERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S)
NAME OF PERFORMING ORGANIZATION: mputer Science Dept. val Postgraduate School; 6b. OFFICE SYMBOL (if applicable): CS	7a. NAME OF MONITORING ORGANIZATION: Naval Postgraduate School
ADDRESS (City, State, and ZIP Code): nterey, CA 93943-5000	7b. ADDRESS (City, State, and ZIP Code): Monterey, CA 93943-5000
NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT ACCESSION

TITLE (Include Security Classification)
velopment of Dynamic, Context-Driven Help Systems

PERSONAL AUTHOR(S)
Nancy Jo McClellan

TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
Master's Thesis	FROM 03/91 TO: 03/93	1993, March 25	128

SUPPLEMENTARY NOTATION: The views expressed in this thesis are those of the author and do not reflect the cial policy or position of the Department of Defense or the United States Government.

COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
ELD	GROUP	SUB-GROUP	Dynamic, Context-Based Help Systems. Dynamic Help Systems. Context-Based Help Systems. Online Help Systems. Help Systems. Computer-Aided Instruction.

ABSTRACT (Continue on reverse if necessary and identify by block number)

This thesis develops a methodology to identify and employ elements of the user's context in the help system architecture, ther roving the response provided by the online help system. Typical online help system structures are static, providing a pre-program onse to a specific assistance request and are not effected by the dynamics of the user or the task being attempted. A dynamic, cont en help system has been developed that uses user- and system-based components of the working environment to influence the sys ess and presentation strategies. The provided response is tailored specifically to the user, based on the user's level of experience and em command history; and specificially to the situation, based on the task being attempted. The resulting online system provides a r ible interface that can serve the needs of all types of users and can evolve as the user's skill with the application grows.

The dynamic, context-driven help system methodology is explained through design and implementation of a prototype help system interactive softltware environment. TAE+ Help is a help component designed to assist users of the Transportable Applicat ironment Plus (TAE+). It is initiated separately from TAE+ but runs concurrently in the XWindows environment. When the user requ stance, TAE+ Help initiates a dialogue with the user, collecting situational environmental information and employs these dynamics in system access process.

DISTRIBUTION/AVAILABILITY OF ABSTRACT: ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED
NAME OF RESPONSIBLE INDIVIDUAL: Timothy J. Shimeall	22b. TELEPHONE (Include Area Code): (408) 656-2509; 22c. OFFICE SYMBOL: CS/Sm

FORM 1473, 84 MAR — 83 APR edition may be used until exhausted. All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE: UNCLASSIFIED

ABSTRACT

This thesis develops a methodology to identify and employ elements of the user's context in the help system architecture, thereby improving the response provided by the online help system. Typical online help system structures are static, providing a pre-programmed response to a specific assistance request and are not effected by the dynamics of the user or the task being attempted. A dynamic, context-driven help system has been developed that uses user- and system-based components of the working environment to influence the system access and presentation strategies. The provided response is tailored specifically to the user, based on the user's level of experience and help system command history; and specifically to the situation, based on the task being attempted. The resulting online system provides a more flexible interface that can serve the needs of all types of users and can evolve as the user's skill with the application grows.

The dynamic, context-driven help system methodology is explained through design and implementation of a prototype help system for an interactive software environment. TAE+ Help is a help component designed to assist users of the Transportable Applications Environment Plus (TAE+). It is initiated separately from TAE+ but runs concurrently in the XWindows environment. When the user requests assistance, TAE+ Help initiates a dialogue with the user, collecting situational environmental information and employs these dynamics in the help system access process.

TABLE OF CONTENTS

I. INTRODUCTION

Despite being the target of considerable attention for the last ten years, there is still little consensus regarding exactly what factors are the critical elements in designing an effective help system. To be effective, a help system must be flexible enough to serve the needs of all users, from the beginning user to the highly experienced. The following sections discuss the basic help system structures and what information is used by the system to generate the assistance response.

A. CHARACTERISTICS OF HELP SYSTEMS

Borenstein [BOREN 85] defines an online or interactive help system to be "computer software that has as its primary function the task of providing the user with information that will assist in the use of some other software system." The typical help system is designed to execute quickly and provides a set answer in response to an explicit user request. The objective is to improve the user's knowledge and proficiency with the application. Variations in the user-help system process include the help mechanism; that is, how the help system is accessed, the question being submitted, the user asking the question and the situation from which the question emerges.

Houghton states in [HOUGH 90] that most help systems are initiated by the user explicitly entering a command-based request - for example "help <command name>". The <command name> is then used by the help system as an index to access the help text database. If the provided <command name> is a valid element of the system vocabulary, the assistance text is displayed; if not, an error message is presented to the user. The help provided usually concentrates on the syntax of the commands, occasionally giving examples, but rarely explains how the command fits into the framework of the whole application or discusses related commands [GWEI 90].

During the system design phase, assumptions are made regarding the amount of computer and topical experience the typical user will bring to the session. Assistance texts are written with this "homogeneous user" [MOILY 87] model in mind and result in a help structure which has both a single retrieval methodology and a preset presentation strategy. Hence, the depth of information provided is invariant and therefore is not impacted either by the individual's experience or ability.

When the help system is initiated, system-related information such as the hardware system in use, the operating system environment and the application being used, is captured. This environmental snapshot

provides most of the system-based influence on the help system access process in a typical help system structure. The user request, containing the command name for which help is requested, provides the only varying component to the help system access equation. The help system builds the help response by combining the system-based framework with the command request and displays the pre-programmed response to the user. If the user requires amplifying information or information on related actions or commands a subsequent help request must be initiated.

B. INFORMATION USED BY HELP SYSTEMS

The typical help system structure does not use user or situation-specific information to tailor the help system access or presentation strategies. The most basic structure, the command-keyword-request structure, is referred to as "static" because the help provided in response to a specific assistance request is pre-programmed and is not effected by the dynamics of the user or the task being attempted [BOREN 85]. However, additional system- and user-based variables, such as application mode in use, the command history of the session and user experience, can be leveraged to provide a dynamic, user-adaptive element to the help system structure. Knowing the application mode being applied provides an additional layer of scope defining information that enables the help system to more closely focus the retrieval. Focusing the retrieval refers to the ability of the system to determine and select for presentation, the help text component that most closely fits the user's help request. For example, if the user is working in a word processing application and wants information about formatting, the help system will have to provide all format-related help text. However, if the application mode in use is known to be "format paragraph", the assistance need can be more closely predicted and the response tailored to provide help information related to the formatting of paragraphs instead of merely defaulting to providing information regarding formatting in general.

Duffy and Langston in [DUFFY 85] put forward one of the more comprehensive discussions regarding employing elements of the user environment in help systems. They discuss the "dynamism" of a help system, or "the degree to which [the help system] can assess and respond to the user's needs of the moment" and further, they state that this is sometimes referred to as "context-sensitivity". The system response is formulated not only by considering the user's explicit request but also by factoring in elements related to the specific user and to the situation that lead up to the request. By employing user- and situation-specific, or context, elements in the help system structure, the help system has a representation of the user environment available to it for use in generating help responses that are more appropriate for the user and the situation.

C. ISSUES ADDRESSED BY DYNAMIC, CONTEXT-DRIVEN HELP SYSTEMS

By definition, a dynamic-based system is more complex to build and maintain than a static help system structure. Therefore, in order to justify the additional effort, some advantage must be offered by the dynamic structure. The following sections address issues involved in considering a dynamic instead of static help system structure.

One of the issues in developing a dynamic-context based help system is determining what types of assistance needs can be answered by a dynamic system that can not be served or served as well by the more typical static system structure. *Assistance needs* addresses both the types of questions the user asks (How do you....?, How can I use....?, What do I do next?, etc.) as well as the type of help that is needed (general, specific, high-level, detailed, etc.). Dynamic structures can assist the user in determining the question that needs to be asked. For example, at times the user may be unable to determine which menu selection will provide the information needed to accomplish the job being attempted. In this situation, the help system mechanism needs to provide a means of prompting or assisting the user, by leveraging the available situational information, to lead them to the question they are attempting to ask.

Dynamic architectures can also support multiple perspectives from which questions may be asked. Each person may think of the task in different ways: one person may think of the task in terms of a particular system function, the next may think of it in relation to the other work being done. The help system must be able to provide a mechanism for the user to use to map the mental and physical pictures together. The way the menu is designed can provide a mechanism to link the user's mental representation of the problem to the help system structure.

Finally, in addition to responding to the explicit request of the user, dynamic structures also have the ability, menu structures that adapt based on the task being attempted, to assist the user in determining the next step in a task sequence. Additionally, they provide an indicator to the user regarding their relative progress in completing the task sequence. The menu structure must be able to provide a logical representation of the how the application is used and should frame the application's commands available to the user and give some indication of their role in the application.

The second issue relates to capturing and employing the user's context for providing more specific help to the user. In this effort, *context* is defined to mean those elements, of both the user model and the system that form a representation of the working environment. Capturing the context entails not only determining how to gather, record and update user and situation elements of the application but also includes determining what elements constitute the context of the application and identify the essential elements of the user model.

Once the decisions regarding how and what to capture are made, the next step is determining how this information can be employed to improve the way the help system access mechanism selects which information will be presented to the user.

Employing the user's context in focusing the help system entry is then a third issue. Once captured, how can the user's context be employed to improve the help system entry mechanism? Static systems typically use only the explicit request and process a pre-programmed help text retrieval. By employing user-based information such as pre-existing knowledge and/or system-based information such as the specific task being attempted, the help text selection process can tailor the response to the user and the user's work situation. Tailoring in this sense means adjusting the amount of information presented relative to the specifics of the user and the work being done.

A final issue addresses formatting the help text to serve users of all levels of experience. The design of the help system is driven by the user model. Therefore the more user-related information that can be captured, the better we can serve the needs of the user. The typical help system development defines the characteristics of the expected/typical user during to the system design phase in terms of the type and amount. A common practice is to present the help information using the same structure, regardless of whether the assistance is being provided to an experienced user or a novice and with no regard as to what type (general or detailed) of information is being requested. A dynamic structure can provide a means of adjusting the information presented to more closely fit the needs of the user, providing more in-depth information to the novice user, for example.

This thesis explores these questions via a prototype help system for an interactive software environment. TAE+ Help is a help component designed to assist users of the Transportable Applications Environment Plus (TAE+). Development of the prototype was accomplished by first identifying the user- and system-based elements of TAE+ that form the framework of the application's context. This was followed by creating mechanisms to first gather and then employ the context-related variables in the help system access and presentation strategies. TAE+ was selected as the target application based on its depth and breadth of the command structure. Additionally, although the current application has a built-in help facility to respond to "what is this?" user queries, many other types of assistance are not provided.

Appendix A provides a synopsis on TAE+ V5.1. TAE+ Help is initiated separately from TAE+ but runs concurrently in the XWindows environment. When the user requests assistance, TAE+ Help initiates a dialogue with the user, collecting situational environmental information and employs these dynamics in the help system access process.

D. THESIS ORGANIZATION

Chapter II of this thesis provides a survey of selected online help systems in general, some of the related criticisms of current systems and a taxonomy to frame the elements of typical help systems. Chapter III describes the methodology employed to develop a dynamic, context-driven help system and the system developed to implement these concepts. Finally, Chapter IV presents conclusions and recommendations for future research.

II. SURVEY OF ONLINE HELP ISSUES

Online help systems are but one element of the much broader online assistance domain that also encompasses online documentation, tutoring, prompting and other such assistance facilities, that element that provides a detailed, localized assistance for the domain of immediate interest to the user. This survey is concerned only with online help systems; specifically, it defines the system taxonomy that will form the basis for describing help systems in this thesis. Since the topic of context is central to the thesis, this chapter discusses the role that context and user experience play in developing a help system separately from the taxonomy even though one potential dimension of the taxonomy would be the use of context. Lastly, this chapter identifies some of the deficiencies of current help systems.

A. TAXONOMY

Help systems can be characterized on a number of dimensions: 1) access initiative, the means by which the help system is invoked; 2) help mechanism, the interface medium between the user and the help system; and 3) help system implementation, the structures used to build the help system.

1. Access Initiative

Help assistance can be initiated by the human user, the system or some combination of both. Systems are classified in terms of access initiative by categorizing them based on the degree to which the system is user- versus system-initiated. The vast majority of systems are user-initiated, that is, help text is provided after a help request is explicitly entered by the user - for example "help <command name>". Systems that supports both user- and system-initiated help are call mixed-initiative systems. An example of a mixed-initiative system, is the WEST system [BURTO 82], which monitors the user's actions and compares the decisions the student makes to the expert module. The help structure provides recommendations for more advantageous moves or actions, as warranted by the user's actions, as well as provides textual help in response to an explicit user request. The final type of help initiation would be a help component that initiates all help assistance. There is no facility by which the user can request assistance; instead the system controls the entire operation and initiates further activity as needed.

2. User-Initiated Help Mechanism

The help mechanism is the interface mechanism by which the user relates to the help system. Possible mechanisms include command-keyword-request, menu-selection, active screen buttons, natural-language request and spoken-phrase request. *Command-keyword-requests* are the most common form of help mechanism ([GWEI 90] and [HOUGH 90]) and require the user to enter "help" followed by the command they are requesting assistance for. Novice users have the greatest difficulty effectively using this type of help system due to the implicit requirement to have pre-existing knowledge about the system and its commands [GWEI 90]. *Menu selection* provides the user with a list of the available help topics from which a choice can be made. This mechanism provides more assistance for the novice by providing a visible list of the available commands but still doesn't assist in the issue of relating the commands to the task at hand and also is often a source of frustration to the experienced user forced through a time-intense access process. A mechanism that is becoming more common is designing user-interface screens with areas on the screen that function as buttons. When these *buttons* are selected (usually by pointing and clicking with a mouse), help texts are shown explaining the item and its purpose relative to the application. This mechanism can be effective for the user "exploring" the application but is less helpful when the user has a specific task in mind and is searching for the appropriate command to accomplish it. Technology in the intelligent and expert system arenas is making advances on providing two additional mechanisms: natural-language requests and spoken-phrase requests. *Natural-language-requests* allow the user to enter a request in the form of a sentence that the system then analyzes [GWEI 90]. *Spoken-phrases* incorporate speech recognition, taking an orally-spoken request and parsing it for keywords. To date, spoken-phrases have been successfully applied only in limited domains [BOREN 85]. At a minimum, systems usually include a command-keyword-request help component and frequently have multiple help mechanisms, for example, command-keyword-request and menu-selection mechanisms.

3. Help System Architecture

Another element of help system development that varies significantly is the amount and type of structural mechanisms that are used in building the help system. Two variants, the utility of which is still being addressed in current literature, are the ideal number of levels of assistance to provide to the user [GWEI 90], [CHERR 89] and [FARRA 92], and whether or not the help system should be part of the application or part of the system as a whole [BOREN 85].

a. Single- versus Multiple-Leveling

A help system structure may be designed to provide a single level or multiple levels of assistance. An example of a single level system is the UNIX "man" (manual) command that provides a copy of the online manual text in response to the user query. The help text may include a recommendation to reference related commands but any subsequent help assistance must be initiated by the user with an additional "man" command [GWEI 90]. Other systems employ a hierarchial menu arrangement of topics and subtopics [CHERR 89] to facilitate navigation through an electronically linked set of help texts. Links may be designed to enable the user to access more detailed texts of the help topic or to access related topic help information. A single level of help provides a simple, easy to understand help structure. Farrand and Wolfe in [FARRA 92] argue that the objective should be to provide simpler and fewer types of help. However, single-level structures sacrifice the power of the multi-level structures. Use of just one additional level provides a mechanism for grouping topics under higher-level titles and reduces the cognitive complexity of the access interface.

b. Integrated and Integral Help System Implementation

Many systems provide several online help system access mechanisms, for example, command-keyword request and menu selection. The level of *integration* refers to the degree of independence the different help mechanisms operarate under. If all system help components access the same database and enable the user to query on any topic independently of the task at hand, it is a fully-integrated help system. [BOREN 85] More commonly, each application has its own help system and a help text database that is accessed only when the user is working within the application.

"An *integral* help system is one in which the help is provided by the same program that executes the command[BOREN 85]. In other words, the help system is invoked from within the application and is actually a part of the application as opposed to being part of the system that runs the application. The advantages of non-integral help systems include providing the user with a consistent interface that is accessible from all applications and often a more comprehensive help mechanism vice the application-specific help.

B. ACCESSING THE HELP SYSTEM

1. Types of User Queries

Sellen and Nicol conducted a number of studies to determine what types of questions users asked. They were able to group the queries into five broad classes of questions as depicted in Figure 1. These classes will be used in subsequent discussions throughout this thesis.

1. Goal-oriented	What kinds of things can I do with this program?
2. Descriptive	What is this? What does this do?
3. Procedural	How do I do this? How do I accomplish a specific operation?
4. Interpretive	Why did that happen? What does this mean?
5. Navigational	Where am I?

Figure 1. Classes of Help Questions [SELLE 90]

In general, goal-oriented and descriptive queries are used by novice users in the process of familiarizing themselves with the application and system. Visibility of the existence of the help component and ease of access are more important aspects than efficiency, as these help components are referenced relatively infrequently by any one user, but may be critically important when they are. Procedural queries are the most common type of queries [SELLE 90] and are used more frequently in general and by all categories of users. Therefore, access mechanism efficiency and ease-of-use are important elements in providing appropriate help information in a format useful to all users. Interpretive and navigational questions relate to *why* and *where* elements in help systems. These help elements are still relatively novel components in today's help system arena.

Two observations related to online help systems emerge: First, "the kind of information delivered should be different depending on the (type of) question asked" (or implied) [SELLE 90]. A user asking a procedural question is likely looking for specific, pointed information, while a request for goal-oriented information indicates more expansive help may be appropriate. Second, "the way in which information is accessed and presented should depend on the question asked" [SELLE 90]. A request initiated by a pointing

action is likely a descriptive query. Contrast this with a command-keyword request. The latter inquiry is likely more procedurally oriented, the former more of a descriptive query. In each situation the type of questions and the manner in which the users ask them are factors that can be employed in the help system design process.

2. Task-Based versus Function-Based Menu Orientation

Jackson, Cherry and Fryer in [CHERR 89] describe a task-based menu structure that maps the order of menu topic titles to the sequence of typical activities that a user must perform to accomplish a major goal. The grouping of titles gives the user a high level view of the whole task and of the sequence of events required to accomplish the work. The objective of this type of menu structure is to assist new users in becoming productive quickly. Additionally, the structure provides an efficient access mechanism for experienced users with a specific request. A function-based menu provides an alphabetical listing of all of the available application commands. Novice users have a more difficulty using this type of help system as it provides no assistance to the novice user attempting to determine which commands are needed to accomplish the task being attempted [CHERR 89].

C. THE ROLE OF CONTEXT

Duffy and Langston [DUFFY 85] define the "dynamism" of a help system to be "the degree to which [the help system] can assess and respond to the user's needs of the moment" and further, they state that this is sometimes referred to as "context-sensitivity". Although the term "context-sensitive" has been used for many years in discussions related to help systems, a universally-agreed-upon meaning has not yet emerged.

Relles and Sondheimer [RELLE 81] describe characteristics of effective online help systems to include "context sensitivity - the ability to provide assistance relevant to the user's current situation". When referring to context, they speak of environmental information related to the user, the application and the tasks being performed. Houghton [HOUGH 90] espouses that the employment of context would be one way to make the help system less disruptive and thus assist the user in maintaining the mental context of the work environment while accessing the help environment. Maass [MAASS 83] declares that systems in general should be able to maintain a "self-image" of their current state and provide explanations to the user about current alternatives and expected formats; this to be accomplished by providing a context-sensitive help system. Sellen and Nicol [SELLE 90] postulate that the more context-sensitive help can be made, the less obtrusive it will be for users.

Burton and Brown in [BURTO 82] describe a expert, computer-based coaching system, "How the West was Won" that gathers the context of the player's game by monitoring the moves the player makes. These are compared to expert module where an evaluation is done to assess the quality of the move.

All systems can be characterized based on the use of context by using a continuum ranging from static help systems, typified by the conventional command-keyword-request system, to dynamic systems like intelligent or knowledge-based systems. The command-keyword-request system is referred to as "static" because the help provided in response to a specific assistance request is not altered by user characteristics or actions. In knowledge-based systems some combination of user experience, the most recently entered command and/or command history is employed by the system to determine what type or amount of help is needed and therefore is characterized as dynamic.

D. USER EXPERIENCE

An additional element is often discounted when discussing help systems: user experience, the amount of both computer and topical experience the user brings to bear in system interactions. Relatively few of the authors substantially address the variation in the type and amount of assistance that is required by an experience-diverse target audience, often stating only that the help system must serve all types of users. The expected proficiency of the user should weigh heavily in design decisions, effecting everything from the amount, type and format of the provided information to the topic organization and access mechanism. Unquestionably, the "novice-novice", a user inexperienced in both computer use and the specific application requires more in-depth assistance than the "experienced-novice", a user experienced with computers but not the application. These needs will also vary significantly from those of the "experienced-experienced" user in search of very specific assistance, for example, command syntax.

E. SHORTFALLS OF CURRENT HELP SYSTEMS

Several authors catalogue/detail deficiencies of current help systems or propose critical elements to consider in designing new systems. Shortfalls that are often cited of help systems include:

- users are required to sift through large verbose help texts for relevant information [BOREN 85]

- users must have knowledge of the available commands, their relevance to the task at hand and the command syntax [GWEI 90].

- users are required to physically as well as mentally switch contexts from the work context to the help system context [RIDGW 87].

- users are provided with a 'canned' help text response, that is, the type and amount of information is not influenced by the state of the user's environment or skill [MOILY 87] and often is too general to be of any assistance.

- users aren't able to determine which commands and actions are needed to accomplish the task being attempted [CHERR 89].

The single message arising from the plethora of help design and implementation articles is that there still is little consensus in the field as to what the characteristics of a good help system are. In fact, Borenstein [BOREN 85] concluded "...help systems in general use are so uniformly unappealing that designers who do make an effort to construct worthwhile help systems tend to assume that they are starting with a clean slate, working on a problem that has never been seriously confronted before...."

Dynamic, context-driven help system improvements proposed in the previous chapter have the potential for relieving many of these deficiencies. The improved access mechanism will result in a more focused retrieval and a response that is more relevant to the situation. Additionally, user-specific information is used throughout the access and presentation processes to tailor the response to the experience level of the user. Improved menu structures will provide a means of presenting a structured command list showing the user not only the commands available for use but also will provide a framing mechanism indicating the role the commands play in the application. By employing user- and situation-specific, or context, elements in the help system structure, the help system has a representation of the user environment available to it for use in generating help responses that are more appropriate for the user and the situation.

F. CONCLUSION

This chapter has presented the user with a taxonomy, or framework, for discussing help systems and has discussed the role of user experience and context in recent help system literature. Additionally, deficiencies cited by displeased users are presented attempting to project the magnitude of the problem. The next chapter discusses the steps necessary to develop a dynamic, context-driven help system and offers a prototype to demonstrate the advantages the dynamic structure offers over more typical help system structures.

III. DYNAMIC, CONTEXT-DRIVEN HELP SYSTEMS

The initial step in developing a dynamic, context-driven help system is to identify the elements of the application environment that form the framework of the application's context. The help system must first gather and track these context elements and then incorporate them into the help system access decision mechanisms to provide an improved help system structure. TAE+ Help has been developed to demonstrate the viability, feasibility and utility of a dynamic-context driven help system. System development excerpts will be referenced periodically throughout this thesis. Appendix A provides a synopsis of the Transportable Applications Environment Plus (TAE+) software development environment application.

A. IDENTIFYING CONTEXT

Context, as defined for use in this thesis, consists of two components, system-based context and user-based context. System-based context elements are identified by actual examination and recording of the system in question; user-based context identification is accomplished by projecting information regarding a typical/hypothetical user of the application. The following sections provide the details of this process.

1. Identifying System-Based Context

Two steps are required to identify and record the elements of a system that define the system-based context. The first step entails developing state-transition diagrams (STDs) to model the system behavior of the application that the help system is being developed for. STDs depict the path(s) through which each state can be reached, the possible transitions out of the state, the catalyst(s) that cause each transition and the subsequent state that results from the transition. These STDs are then employed in the second development step to isolate/identify system functions and actions supported within each state. Appendix B provides STDs generated to model applicable portions of TAE+. TAE+ Help STDs associate a state with each TAE+ window/subwindow.

Each STD state is examined individually to identify and record the questions the user may ask/ have while working in the state. Sellen and Nicol in [SELLE 90] define five types, or classes, of help-related questions users typically ask: goal-oriented, descriptive, procedural, interpretive and navigational. A table is constructed that cross references, by question class, the questions discovered in each state and the STD state they apply to. In addition to the organizational benefit, this question-state table construct provides a check-

and-balance mechanism to ensure comprehensive coverage of the state assessment process. The developer first lists the questions that arise after the initial review of the state and groups them according to the five-class structure. Then the content of the resulting question-state table is reviewed from the perspective of each class of question, to ascertain all probable questions have been included. The content of the question-state table then becomes the foundation document for development of a three-tiered menu system that will be used to gather the user's working context. Appendix C provides the question-state table containing the questions that arise at each TAE+ state. TAE+ V5.1 supports the descriptive questions within the current application and thus they have not been included in this effort.

A second table is constructed for recording visible objects, or observables, such as subwindow(s) titles, icons, operating mode setting, etc., within the state that are state-specific. The observables-table structure also includes developer notes regarding supplementary questions that may be asked of the user to more closely pinpoint the context elements where there remains a lot of context ambiguity. Appendix D provides a table of the observable context clues associated with each TAE+ state.

These two table structures represent two of the system-based context elements that must be captured and employed in the developmental help system. The final system-based component is the application mode in use. Typically, the application mode categories offer a high-level, system-partitioning mechanism that can be used to quickly reduce the search list of potential topics that must be considered during the help system access process. If the application mode being applied is known, entire sections of the system can be dismissed during the initial topic honing effort, leaving a smaller, more manageable quantity to address in subsequent searches. The more closely the target topic can be pinpointed during the access process, the more request-applicable the information that can be provided to the user. The next component that is addressed is identification of the user-based context elements.

2. Identifying User-Based Context

User-based context elements, when considered together, form the user model that will be assumed throughout the development. The model should profile the typical user of the application; in the case of a help system, this still infers a very broad range of capabilities/knowledge that must be provided for. Elements to consider include user experience and help-system-command execution history.

a. User Experience

The amount of previous experience the user brings to the session contributes significantly to decisions regarding how much help information is provided to the user in reply to a request for system help

and the manner in which it is presented. For example, novice users need more intuitive menu selections to choose from and in the case of requests for very specific/narrow scope information, it may be advisable to provided additional assistance in the form of more general topic help. Also, previous experience may not necessarily be directly related to the target application in order to be considered significant. An additional type of experience available for use in the access decision process arises if the topic in question in some way relates to another topic that the user may be familiar with. In this instance, the provided help text can be presented in terminology relating the known topic to the requested topic, assisting the user in using pre-existing knowledge to interpret and understand the new situation.

b. Menu Selection History

Topic selection and the history of topics selected by the user are also important elements in the help system access process. The user's topic selection itself indicates the explicit user request and may also imply the application development stage that the user is currently working on. However, the history of topics selected implies additional information about what assistance the user may need. For example, if the user is a novice and has not requested or been shown general information about a concept, responding directly to a very specific topic request may not be adequate. It may be advisable to preface/supplement the specific topic response with a broader topic discussion segment to clarify its full role in the application.

B. GATHERING CONTEXT

Once the meaningful elements of the context are identified, they must be collected and tracked for use in the help system access decision process. System-based context elements are gathered by recording the application mmode and menu/submenu selections made by the user. User-based elements are gathered by monitoring the skill level and past experience of the user.

1. Gathering System-Based Context

A three-tiered menu system is used to create a multi-prospective view of the target application system for the user. By monitoring the menu topic selections made by the user a context model can be constructed and used as the basis for determining what topic the user is interested in.

a. Subwindow Menu

The highest-level menu structure, the *Subwindow Menu*, is developed from, and ordered by, the hierarchy of the application's subwindow titles. The user selects the menu title that corresponds with the title of the application subwindow being worked in. Embedded submenus prompt the user for more

information regarding their help request by projecting lists of probable topics and thereby provide a mechanism for capturing more of the user's working context. The *Subwindow Menu* provides the user with a very efficient help access mechanism, directly mapping the question-related application window to the help structure and quickly focusing the help topic lists to the probable topic area. Supplemental submenus, embedded within the subwindow structure, enable remaining topic ambiguity to be further reduced.

b. Task-Oriented Menu

The second tier, the *Task-oriented Menu*, is ordered based on the sequence of tasks accomplished during a typical development. The user selects the menu title that corresponds with the task currently being worked on or for which information is being requested. Supplementary questions may be asked to determine if the user has any other direct/indirect experience that may be leveraged to best answer the current request. The user is able to relate the task being accomplished directly to the menu structure, providing an indicator of what the user's working context is now, what the next context is likely to be and allows suppositions to be made regarding what the user has knowledge of. For example, in TAE+ Help, if the user is currently working on specifying an item, it can be assumed that the tasks related to resource file manipulation and panel specification have already been accomplished by the user and it is probable that help requests in the near term will relate to the details of building and placing items.

c. Function-Oriented Menu

The final tier, the *Function-oriented Menu* consists of an alphabetical listing of the projected system functions that have been derived from the question-state table contents. The user selects the menu title that best represents the topic for which help is being requested. The function-oriented listing is the finest-grained topic listing, providing the building blocks to represent all the atomic/fundamental context elements of the application. Whereas the subwindow and task-oriented structures assist in the capture of the physical working context of the application, the function-oriented structure provides a capability for capturing the user's mental context of the problem in situations where the request is unrelated to the current physical context.

The three-tiered menu system provides the user with a help system configured in multiple fashions, enabling the user to map his mental picture to the most appropriate help view, and thereby reducing the user's cognitive load when moving between the work and help environments. The subwindow and task originated requests use their respective structures to support the gathering of the context-based elements of the situation

and pinpoint the primary topic title. System control then passes to the function-oriented structure to present the requested help text. Access to the menu structures is provided to the user through left-, middle- and right-button mouse clicks to reach the Subwindow, Task-oriented and Function-oriented Menus respectively.

d. Application Mode

The application mode setting information is gathered during the initial system entry process and again whenever the user's topic selection indicates the application utilization has moved into a different mode. In TAE+ Help, the mode the user starts working in is the Move/Resize/Edit mode. Menu lists are ordered to first present the topics expected to be encountered by the user working in that mode. If the user subsequently requests assistance on defining Panel connections this is an indicator that the application mode has changed or should be changed to Define Connections. If, in fact, the development has progressed to the point of defining connections between objects and panels, a further supposition can be drawn that the topic lists should be reordered to first present titles related to defining connections and other tasks that fall after that point in a typical development effort.

2. Gathering User-Based Context

User-based context element information is gathered initially upon entry into the system and then is updated dynamically throughout the session. Contributing elements include user experience and command history.

a. User Experience

The initial experience level setting is based on responses to an experience assessment conducted when the user first enters the help system. Questions designed to break users out into distinguishable levels of experience are presented and the responses are used to derive the assigned experience level. In the case of TAE+ Help, the user is queried as to the extent of their experience in the use of the more complicated TAE+ capabilities. Their replies are then used to classify them as novice, experienced or knowledgeable users. Subsequent application use adjusts this proficiency variable dynamically to more closely map the demonstrated user skill to both the amount and type of provided information. For example, if the user replies indicate they have experience with several of the more advanced development procedures, the user is classified as Knowledgeable. However, if a user classified as Knowledgeable selects very fundamental menu topics, the user will be re-classified to downgrade the

assigned experience level to Experienced. Conversely, a user classified as Novice will automatically be upgraded to Experienced after accessing the post-TAE+ Item topics.

b. Menu Selection History

The command selection history together with the experience level of the user are the fundamental factors in the help system access decision process. The command history is accumulated by flagging each help script element that is viewed by the user. In TAE+ Help, a topic is flagged as *shown* after being viewed by the user. The experience level of the user, along with the list of *shown* topics, contribute to the decisions regarding what the user sees in response to the current request and when the experience level of the user is modified/adjusted. If the user is a novice user and is requesting assistance on a very specific topic, a check is done to see whether the user has viewed the more general topic script. If not, the general information is presented and is followed by the specific topic requested by the user. Further details of this process are provided in the following sections.

C. USING CONTEXT

The gathered context elements are used and updated throughout the application session. The organization of the menus and user experience, direct or indirect, are the primary context driven components that evolve as system utilization progresses. The following sections provide the details of the how context is employed in the developmental help system.

Appendix E provides a program listing of TAE+ Help. It has been developed using a special-purpose grammar, interpreter and parser created by D. Maskell [MASKE 92]. Page 2 of Appendix E provides an example describing the structure of the state-based grammar. Appendix F provides this grammar in Backus-Naur Form (BNF). In general, the format is as follows: in the program listing each code segment defines the state being specified, the action(s) that result upon occurrence of the state and the catalysts that cause a transition and associated actions that result from invocation of the catalyst.

1. Using System-Based Context

a. Menu Access Mechanisms

The application mode being employed by the user is used as a partitioning mechanism to tailor menu access points, providing the menu page to the user that contains the topic titles associated with the application mode in use. For example, if the current WorkBench Mode is Define Connections, it is probable that the user is searching for Panel-related help. Therefore, the menu page that should be provided

first should be the page that contains the Panel-related topic titles. Conversely, if the current WorkBench Mode is Set Item Default, it is probable that the user is searching for Item-related help. The page that contain the Item titles should be presented first.

b. Menu Transition Mechanisms

The underlying premise of the menu/submenu transition is that presenting topics on the new menu page that are most-like the topics on the page that was left, helps the user maintain the mental context of problem request and directly presents "most-probable" topics for further selection. In TAE+ Help this occurs in both main menus (Subwindow, Task-oriented and Function-oriented Menus) and submenus (within the Subwindow Menu structure) to main menus.

Code excerpts from Appendix E are provided in Figure 1 below to demonstrate the menu-to-menu transition process. (An explanation of the grammar is detailed on page 2 of Appendix E.) The following code segments are from states st_1_3_300 and st_1_2_150. While in state st_1_3_300, page 2 of the function-oriented menu structure, if the user clicks the left mouse button (click-left), control transfers to state st_1_2_150, the subwindow page containing topic titles most-similar-to those found on the st_1_3_300 page.

```
(st_1_3_300,
((0, clear),
  (0, write,"16. Exiting from a resource file      23. Item
             17.  with save                        24.  Aligning multiple items
             18.  without save                     25.  Creating
             19. Exiting from TAE+                 26.  Data constraints/null value
             20. Generating source code            27.  Data Driven Objects (DDOs)
             21.  Single/multi file generation     28.  Default value
             22. Including resource file w/i       29.  Deleting item
                 another resource file             30.  Duplicating item(s)
  (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented
topic lists respectively."),
  (9, input, mouse&key)),
      (extraneous code removed here...)
((click-left, st_1_2_150),   /*left button calls for sub-w list*/
  (click-middle, st_1_3_1), /*middle button calls for function list*/
  (click-right, st_1_4_300),  /*right button calls for task list*/
      (extraneous code removed here...)
  st_1_3_150))


(st_1_2_150,
((0, clear),
(0, write, "8. PANEL                    12. ITEM
            9. Panel Specification      13. Item Specification
            10. Panel Details           14. Item Constraints
            11. Dimension Specification 15. String type constraints
                                        16. Integer type constraints
                                        17. Real type constraints
                                        18. Dimension Specification
                                        19. Presentation Details
    (extraneous code removed here...)
(Enter topic number, 'N' for next subwindow page or 'P' for previous subwindow page)"),
("n"|"N", st_1_2_200),
("P"|"p", st_1_2_1),
    (extraneous code removed here...)
st_1_2_375))
```

Figure 1. Menu-to-menu Transition Process Example

2. Using User-Based Context

a. Setting and Modifying User's Experience Level

The initial setting of the experience level is derived during the system entry from question/ responses geared to provide clear separation points between the experience levels. Subsequent application use modifies the initial setting based on help system access requests. TAE+ Help supports three experience levels

but any number of distinguishable levels can be supported. The code segment provided in Figure 2 sets the experience level to Knowledgeable. State st_1_1_4 queries the user regarding his experience with the application. If the user has had experience with the more advance functions, the user's skill level is assessed to be Knowledgeable and control transitions to st_1_1_300 to accomplish the actual setting. In state st_1_1_300, after clearing the screen, previous settings (if any) are nullified by the *retract* action. A message is provided to notify the user of the experience setting and the level is established by the *assert* action prior to transiting to the next state (See Appendix E, states st_1_1_100, st_1_1_200 and st_1_1_300 for the novice, experienced and knowledgeble setting mechanisms, respectively, and st_1_4_3 - st_1_4_11 and st_1_4_23 - st_1_4_47 for modification mechanisms).

```
(st_1_1_4,
 (0, write, "Have you used either the TAE+ Command Language (TCL) to update objects or
detail TAE+ generated source program event stubs? (Please enter Y or N)"),
 (9, input, keyboard)),
 ((15 seconds & (past st_1_1_4 wait), st_1_1_25),
 (15 seconds, (assert, wait), st_1_1_4),
 ("YES"|"Yes"|"yes"|"Y"|"y", st_1_1_300),   /*set experienced level to Knowledgeable*/
 ("NO"|"No"|"no"|"N"|"n", st_1_1_5),
 st_1_1_4)) /*unexpected ans*/


(st_1_1_300,
 ((0, clear),
   (0, retract, st_1_1_100, novice_user), (0, retract, st_1_1_200, exper_user), (0, retract,
st_1_1_300, knowl_user),
   (0, write, "...experience level set to KNOWLEDGEABLE..."),
   (6, write, "please click the mouse...")),
   /* from 1_4_47 to 1_4_49 upgrade exper level from exper_user to knowl_user*/
 (((past st_1_4_47 active), (assert, knowl_user), st_1_4_47),
   ((past st_1_4_48 active), (assert, knowl_user), st_1_4_48),
   ((past st_1_4_49 active), (assert, knowl_user), st_1_4_49),
   /* No active states during the initial setup - falls thru to here */
   (assert, knowl_user), st_1_1_6))
```

Figure 2. Initial Experience Level Setting Example

The code segment provided in Figure 3 below is excerpted from state st_1_4_5 to demonstrate the mechanism used to downgrade experience levels when topic selection indicates an inappropriately high experience leve setting. A knowledgeable user is not expected to require assistance on fundamental topics like opening the resource file. When this topic is selected and the experience level of the user is *Knowledgeable*, an *active* flag is set and control passes to state st_1_1_200 ((past st_1_1_300 knowl_user),

(assert, active), st_1_1_200). State st_1_1_200 resets (downgrades) the experience level to *Experienced.* When st_1_1_200 concludes its action, it looks for an active flag. Finding state st_1_4_6 active, control returns. Now, redesignated as an experienced user, the requested topic will be presented to the user ((past st_1_1_200 exper_user), (assert, shown), st_1_3_61).

```
(st_1_4_5, ((0, clear), (0, retract, st_1_4_5, active), (6, write, "please click the mouse...")),
 (((past st_1_1_300 knowl_user), (assert, active), st_1_1_200),
                                        /*Knowledgeable shouldn't be asking this question*/
   ((past st_1_1_200 exper_user), (assert, shown), st_1_3_61),
                                        /*Non-novice sees specific info directly*/
   (((past st_1_1_100 novice_user) & (past st_1_3_60 shown)), (assert, shown), st_1_3_61),
                                        /*Novice user shown genl info, then returned to*/
                                        /*show specific info OR novice saw on previous req*/
   ((past st_1_1_100 novice_user), (assert, active), st_1_3_60)))    /*Novice needs to see genl
info*/
```

Figure 3. Modifying the Initial Experience Level Setting Example

b. Using Experience in Access and Presentation Decisions

(1) Experience level. Experience levels are used in decisions regarding how much information to present to the user and the manner of the presentation. In TAE+ Help, the initial experience level setting is derived from past experience that the user brings to the use setting. The initial setting is then updated throughout the session by tracking what has been *shown* to the user prior to this point and what topic is being currently being requested by the user. Figure 4 depicts the help system assessment and access process. Upon receipt of a request, TAE+ Help reviews the user's experience level in light of the request and a determination is made as to whether the current experience level setting is appropriate. The existing experience level may be up- or downgraded by the system based on the user's topic selection. After the assessment, the experience level is then used to determine what help text should be presented to the user. *Novices* are shown both the general and specific topics' help scripts if the general information has not been shown prior to the request being addressed. For knowledgeable, experienced or novice users that have been shown the general topic help prior to the current request, presentation moves directly to show the specific text script (See Appendix E, st_1_4_38 and st_1_3_23/st_1_3_34).

(2) Using pre-existing knowledge. The typical user can be expected to have some other computer experience and may even have some amount of experience with the target application that may be

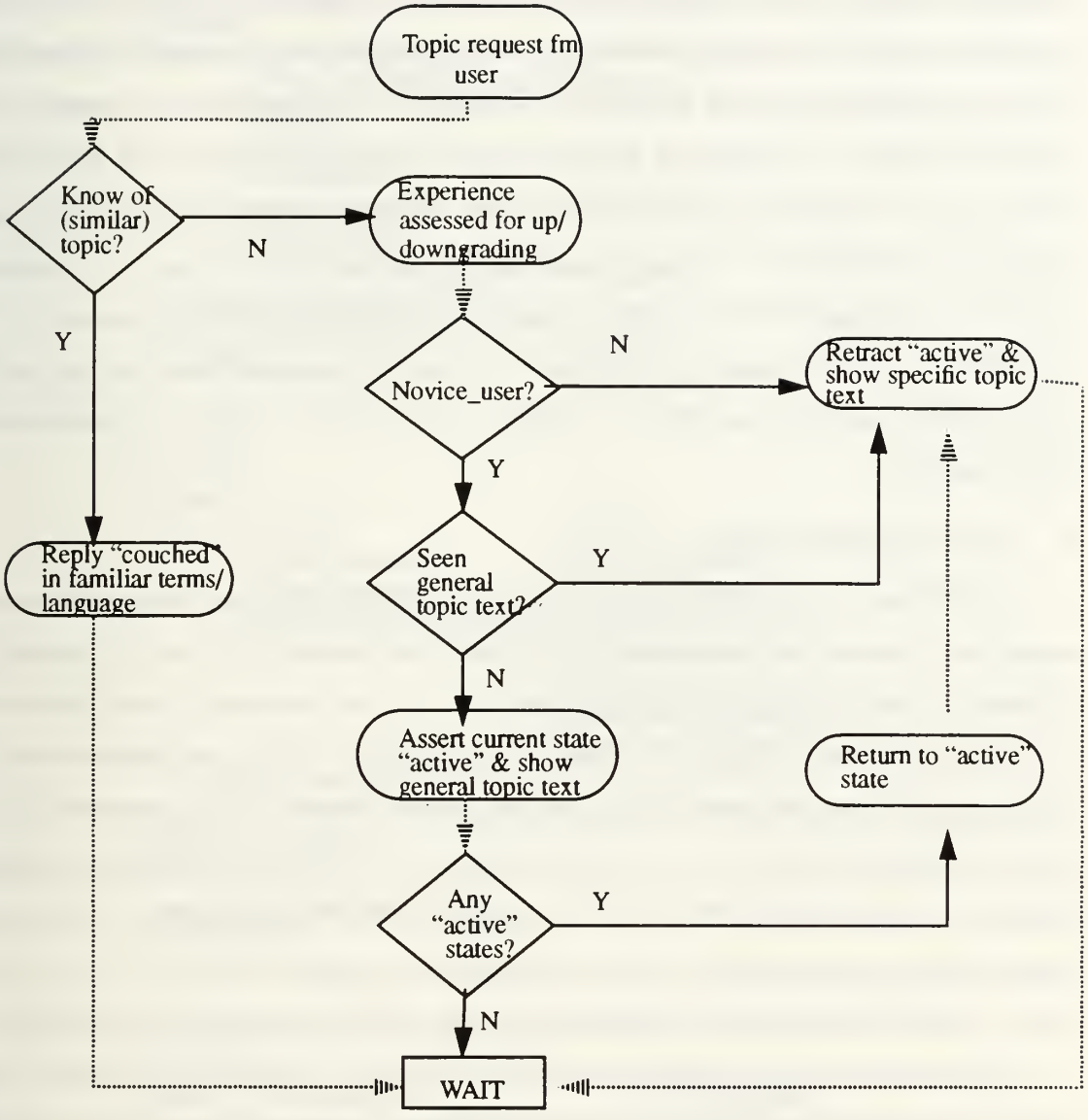


Figure 4. Employing User Experience

similar to operations being documented in the help system. If two operations are known by the developer to be similar in some way, the help text may be written to relate the two operations. In this case, the user would be queried as to if they have had any experience with the other topic. If so, the new topic is presented using the somewhat familiar terminology. Otherwise, no assumptions are made and the topic is presented in the more typical fashion. For example, in TAE+ Help, when a user requests help for Data Driven Objects (DDOs), the user is queried as to whether they have any experience with other types of dynamic items, specifically, Selection Objects. If the answer is affirmative, the requested DDO help text is explained in terminology that leverages the existing Selection Object knowledge and thereby reduces the amount of "new" information the user has to contend with. If the answer is negative, no assumptions are made and the DDO topic is explained fully (See Appendix E, st_1_4_300, st_1_4_28, and st_1_4_28_2 for the implementation of this capability).

D. THE TAE+ HELP SYSTEM

In TAE+ Help, a three-tiered menu hierarchy system leverages and updates the gathered contextual information to facilitate the help system access process. The first menu tier is the *Subwindow Menu* component. Its topic list consists of a collection of the titles of the TAE+ subwindows that the user sees while accomplishing the typical application development. It is intended to support queries related directly to a known subwindow. Upon selection of a title by the user, submenus are generated and presented to the user to facilitate more precise determination of the user's question prior to selection of the topic text component assessed to be most applicable. Once the topic is pinpointed, system control transitions to the function-oriented program component and the help text is presented (See Appendix E, states st_1_2_1, st_1_2_150 and st_1_2_200 for the full *Subwindow Menu* implementation. Excerpts are provided below in Figure 5.).

The *Task-oriented menu* component, the second tier, provides a collection of topics, ordered by task name. The ordering of titles within this menu is based on the order the user would encounter/accomplish tasks when progressing through the typical interface development. It is intended to support users needs regarding "Where am I?" and "What's next?" in the development process and possibly "What can I do?". Within this component, user experience classification is checked to reconcile the request with the recorded experience level and any experience level adjustments are made as deemed appropriate. Additionally, decisions on the need to provide broad topic information text in addition to the requested specific topic are made based on the experience level of the user and the history of help received prior to the current request. If the user is a *Novice* and is asking for specific topic information, but has not yet seen the general topic help text; the current state

```
(st_1_2_1,
((0, clear),
(0, write, "1. Sub-Window List
            2. TAE+ WorkBench: Resource Filename
            3. FILE
            4. New File
            5. Open File
            6. Include File
            7. Save As File
            8. PANEL/ITEM/OTHER COMMANDS
    (extraneous code removed here...)
("8"|"N"|"n"|"Eight"|"eight"|"EIGHT", st_1_2_150),
st_1_2_225))


(st_1_2_150,
((0, clear),
(0, write, "8. PANEL                    12. ITEM
            9. Panel Specification      13. Item Specification
            10. Panel Details           14. Item Constraints
            11. Dimension Specification 15. String type constraints
                                        16. Integer type constraints
                                        17. Real type constraints
                                        18. Dimension Specification
                                        19. Presentation Details
    (extraneous code removed here...)
(Enter topic number, 'N' for next subwindow page or 'P' for previous subwindow page)"),
("n"|"N", st_1_2_200),
("P"|"p", st_1_2_1),
    (extraneous code removed here...)
st_1_2_375))


(st_1_2_200,
((0, clear),
(0, write, "20. DUPLICATE                24. AUXILIARY
            21. Group Modification      25. Specify Initial Panels
            22. ALIGNMENT               26. Application Generation
            23. Current Selection Alignment 27. Terminal
                                        28. WB Preferences
                                        29. Connections Specification
(Enter topic number or 'P' for previous subwindow page)"),
(extraneous code removed here...)
```

Figure 5. Subwindow Menu Implementation Code Segments

is flagged as *active* and system control transitions to the function-oriented program component where the general topic is presented. Then control returns to the *active* state and the specific topic request is processed (See Appendix E, states st_1_4_13, st_1_3_43 and st_1_3_44 for an example of this implementation in TAE+ Help).

Finally, the *Function-oriented Menu* component provides an composite listing of topics, ordered alphabetically. This list includes all of the *Task-oriented Menu* topic titles plus any other topics that can be expected to arise from previous computer experience (e.g., use of word processing packages, other development or graphic tools, etc.). This topic list provides the "finest-grain" level of topic selection and is intended to support "How do I?" queries, enabling the user to quickly find the means of accomplishing a known action/capability. The function-oriented program listing section is where all help texts are coded. Subwindow and task-oriented menus pinpoint the topic title and access the applicable function-oriented topic script for the actual help text (transparent to user).

Within each help system window, the user is given access to left-, middle- and right-mouse-button clicks to move from one menu structure to another. Menu and submenu page linkages transit to similar-topic pages within the other menu structures. This reduces the user's cognitive load thereby assisting them in focusing on maintaining the working context vice worrying about the mechanics of the menu system. For example, the transition out the subwindow menu page that has TAE+ Panel-related titles on it results in accessing the task-oriented menu page with Panel titles on it. A left click invokes the *Subwindow Menu*, a middle click the *Task-oriented Menu* and a right click the *Function-oriented Menu*.

E. CONCLUSION

This chapter has presented a description of the development process used to identify, gather and employ context in the access and presentation strategies of a dynamic, context-driven help system. Additionally, the chapter provides an overview of TAE+ Help, a help system prototype developed to demonstrate the advantages afforded by integrating user and system elements into the help system structure. The last chapter summarizes the utilization of context in the prototype and discusses directions for further research.

IV. CONCLUSIONS AND RECOMMENDATIONS FOR FUTURE RESEARCH

The purpose of this thesis is to develop a method of identifying, capturing and employing elements of the user's working context to create a dynamic, context-driven online help system. This chapter provides a description of the help system development process, a summary of the contributions and discusses recommendations for future research.

A. METHODOLOGY OF PROTOTYPING A HELP SYSTEM

A help system prototype has been developed to demonstrate the improved help system effectiveness afforded by integrating and employing system- and user-based elements in the help text retrieval process. The initial effort involved defining the user model as well as a strategy for revising the initial user model as the user's experience evolves or as help title selection indicates is warranted. The development of the help system decision structure occurs next and relies heavily upon the dynamics of both the user and the situation.

The advantage of basing the help system's user model on both experience and situation include an ability for the help system to adapt in two dimensions. The help system mechanism evolves as the user's experience grows, changing from prefacing specific requests with general, framing assistance to directly provide the requested assistance. The mechanism could also be used to support further tailoring of the presentation to provide additional, specially formatted information for the more advanced user. The help system mechanism also evolves based on the application mode being applied, by tailoring the menu structures to present and access the topic titles most appropriate to the current work situation.

1. The Initial User Model

The initial user model was evolved based on two components: user experience and application mode being applied. When the help system is entered, the user is queried on the experience they have using the application. The questions are formulated to provide a partitioning mechanism that affords a means of categorizing each user into a single experience level, for example, novice, intermediate, experienced, etc. Each user is rated upon entrance into the help system and that rating serves as the experience level variable in the help system access process until it is changed by further help system use. The second component is the application mode being applied by the user. The mode provides an additional source of scooping information that is then used to improve the relevancy of the information that is provided to the user. The user is queried

as to whether the application mode has changed if the help title being selected indicates that the mode has or should be changed. These are the elements that are updated periodically throughout the session and provide the mechanism by which the interface adapts.

2. Strategy for Using and Revising the User Model

The user experience level influences the order and presentation of the help text. If the user is requesting assistance on a specific topic and is an inexperienced user, the help command system history is consulted to determine what information has been presented to the user during the session. If the user has not been presented with the general topic, it is assumed that the user would benefit from the additional background information and results in the presentation of the general topic followed by the specific topic text. The experience level is up- or downgraded throughout the session by monitoring and assessing the help topic title selections in conjunction with the assigned experience level. For example: if the user is assessed to be an inexperienced user upon system initialization they be provided with general topic information before the specific, then once they complete the tasks up to a pre-determined point in the application, the experience level automatically adjusts to reclassify them as an intermediate user. Downgrading occurs if the topic selections being made indicate that the experience rating was set inappropriately high.

The advantage of the dynamic structure over static help system structures is the ability of the user model to adapt based on the user's experience. By revising the user model, the help system is able to adapt to the user, not just within the session but also throughout the lifetime that the application is used by the user. The first time the application is used, the user will probably be assessed as a novice user. The experience level assessment mechanism is flexible enough to adjust the user's experience level during the session as the user becomes more knowledgeable (or is deemed to be less knowledgeable than assessed). Additionally, upon subsequent entries into the application previous experience can then be factored into the assessment and may result in a more advanced experience level setting. Static structures are invariant. They provide the same response to the assistance requests, with no influence exerted by the user's experience.

3. Developing the Help Structure

The development of the help structure is accomplished by first modeling the system behavior using state-transition diagrams (STDs) and from that identifying the types of help questions and the context indicators that are evident at each state. The questions and context indicators become the foundation from which the menu structures and help text presentations are formed. This is advantageous as it facilitates an incremental and more comprehensive development effort. The tables are built through examining each state.

The resulting tables are used directly to build the three menu structures and form the text presentation decision structure. The subwindow and task-oriented menu structures fall out of the association of topic titles to application states that the tables portray. The function-oriented menu is merely a superset of all of the subwindow and task-oriented menu titles plus any additional titles that may result from pre-existing computer or other application-use experience.

a. Identify the Situation

The time dependent behavior of the system is captured by recording possible states that occur in the system, the paths used to reach the state, the possible transitions out of the state and the actions resulting from the transition. Yourdan's STD structure, described in [YOURD 89], is used to depict the application's behavior.

Once the STDs are complete, each state is assessed in isolation to note and record help questions that are prompted by the state. The questions are organized into the five question-class structure described by Sellen and Nicol in [SELLE 90] and the resulting table is used to cross-check the completeness of the state assessment. The advantage to this approach is this provides a tool for creating a help system that attempts to anticipate all of the assistance needs that any user, from the novice to the knowledgeable, may have while using the system. Focusing on each state individually provides the developer with a logical way to partition the system and facilitates identifying all questions that arise within the state. The resulting table also provides a way of identifying the different perspectives from which each question might arise.

b. Identify Relevant Topics/Questions

A second table structure is developed with the assistance of the STDs to identify and record "observables" such as window titles, visible icons, etc., specific to the state that can be used to collect situation-specific context data. Additionally, as the states are reviewed, questions are recorded to be put to the users in situations where context ambiguity might be further resolved. These observables allow the help system to better focus the access, providing information that is more relevant and specific to the user's situation.

c. Compose the User Message

The third phase of the help system development uses all of the previous user and system mapping effort to create the interface itself. Menu and submenu structures are developed from the five-class questions and the context observables tables; menu ordering and transitions fall out of the STDs and the text

script and presentation details emerge from the user model projections. The resulting interface provides a help system that adapts to the user throughout the session, and indeed throughout the lifetime use of the application, by capturing and recording context-related information from the working environment and employing it in the help system access process.

B. SUMMARY OF CONTRIBUTIONS

A dynamic, context-based help system uses snapshot representations of the work environment situation and the user to create a help system architecture that evolves throughout the session. The resulting assistance system uses the situational elements to predict the topic of the question being asked thereby 1) assisting the user in forming the question and 2) improving the help system access process. This results in selection and presentation of help text which is more relevant and specific to the work situation.

Selected topics can be related to similar topics, whether they relate directly to another function of the application or relate to some other common activity (like saving a file, for example). When this is possible, the user is first presented with a question asking if they have knowledge of the related activity. If so, the new topic is presented to the user using terminology to relate the new to the known topic, lessening the amount of new information the user must conceptualize.

The user elements are used to tailor the response to the specific user by determining the sequence and amount of help assistance that should be provided. Less experienced users are presented general assistance as well as the more specific text to improve their base of information of the application's commands. The tailoring mechanism can also be used to modify the presentation of experienced users by providing additional shortcut key sequences/hints for more efficient usage. Due to limitations of the grammar used for implementation, this capability has not been provided in the prototype.

The final element of the dynamic structure that was improved is the menu structure. Three topic list structures are supported: a subwindow menu organized by application subwindow title, a task-based menu organized by sequential ordering of the typical tasks used and a function-based menu organized alphabetically by topic. The three-tiered menu organization provides several mechanisms for the user to chose from in mapping the question-to-be-asked to the help architecture. The subwindow structure provides a one-to-one direct mapping of the application subwindow to the help system menu. The task-based list provides a link between the task being attempted as well as indicates the next logical task and the relative progress made in completing the task sequence. Lastly, the function-based list provides a quick access mechanism to locate assistance for a known capability.

C. RECOMMENDATIONS FOR FUTURE RESEARCH

This thesis concentrated on developing a methodology for identifying and employing elements of the user's context to improve assistance provided by online help systems. The early developmental process entails diagramming and examining the target application's functionality to identify the context-framing elements of the situation. Done manually, this is a tedious, detail-laden process. A automated tool is needed to assist the user in identifying and verifying the completeness of the context-assessment effort. The tool could verify that all state-transition diagram state paths have been investigated and that all classes of user-questions have been addressed.

Another piece of the development process involves identifying questions that may be asked of the user to supplement the help-system-gathered context information. The current system configuration accomplishes the context gathering via a manual, time-intensive question-and-answer dialogue between the user and the menu structures. An automated context sensor mechanism, and indeed a theory of designing sensors, needs to be developed to collect situational data electronically, in a way that is transparent to the user.

One of the presentation strategies of the dynamic help system structure is to exploit the user's experience with similar tasks. Adding additional strategies provides more interface flexibility, such as presenting related commands or keyboard shortcuts to experienced users, but it is difficult to add to an existing dynamic help system structure. Changes ripple through the system and can cause both unanticipated and undesireable changes. The current manual process requires the developer to manually trace all of the state-transition paths for affected states and then retrace each again if further changes are added. This tracing quickly becomes unmanageable as the system functionality grows. An automated tool to display and manage the help system structure is needed to provide the developer with a pictorial representation of all of the affected paths and states. Additionally, a methodology is needed to efficiently add or change strategies or other elements of the user or system models.

A strategy is needed to manage or better exploit the knowledge that the system has and acquires about the user. An assumption is made that once the user has been *shown* a help topic text they understand the topic and how it fits into the framework of the application. A mechanism is required to verify the user's understanding of the provided assistance. Additionally, this mechanism should feedback information to the sensor structure for use in monitoring the user actions and assisting in subsequent issues like whether additional background information on a specific command/action is necessary. Finally, the issue of how many levels of experience to design into the help system structure was addressed ad-hoc in this thesis. Additional research, to provide a solid basis for this decision, would be useful.

APPENDIX A

TRANSPORTABLE APPLICATIONS ENVIRONMENT PLUS (TAE+)

A. INTRODUCTION

Transportable Applications Environment Plus (TAE+) is a portable software development environment that supports rapid building, tailoring and management of graphic-oriented user interfaces. It uses MIT's XWindow System (Ver 11, Rel 4) and the Open Software Foundation's (OSF's) Motif Toolkit. It can generate source code in C, C++, Fortran and Ada as well as high level TAE Command Language (TCL).

B. OVERVIEW

This section provides a description of the TAE+ working environment and one of its major components, the TAE+ WorkBench, thus forming a conceptual framework within which this thesis it to be read.

1. TAEnvironment

TAEnvironment is an encompassing environment that manages and provides access to the TAE+ development tools via an icon menu. Components include: the WorkBench; taeidraw, a graphic drawing tool; windows for access to the host operating system and FTP; Code Generator; a utility program that generates source code from the TAE+ resource file, and an object bank and demo component which provide interactive samples and implementation of a variety of TAE+ objects. This thesis applies primarily to the TAE+ WorkBench and its utilization by the developer to accomplish the fundamental development of an interface, to wit, the visual appearance, the user interaction mechanisms and the intra-interface linkages and actions.

2. TAE+ WorkBench

The TAE+ WorkBench is a developmental tool that supports the interactive design and layout of graphical user interfaces [NASA 91]. There are two main elements used in building the interface: *panels* and *items*. *Panels* are the windows seen by the application user. In the [context - need correct word here] of TAE+ they hold a collection of interactive *items*. *Items* are the mechanisms by which interaction (input and/or output) between the user and the application is accomplished. Examples of *items* include buttons, text display boxes, scroll bars, menus, etc.

The first step in the development effort is to create the visual picture of what the user sees when using the application. Specification of panels and items occurs while the TAE+ WorkBench is in "Move/ Resize/Edit" mode. Figure 1 provides a sample electronic day planner page to support this explanation.

a. Create Panels

The developer first creates a panel and specifies the desired physical characteristics of the panel. Typically, the panel size and screen location are defined as well as the desired font and screen color.

b. Create Items

Next, the developer creates and specifies the items which will be visible on the panel. For our day planner page, appropriate items might be: a calendar month display area, buttons to provide QUIT, and ENTER NEW, MODIFY, DELETE of existing appointment information and Text display/label items (see Figure 1). Creating and specifying each of these types of itemsis address individually below.

To create the calendar month display area, first create a new item, specifying an unique item name and defining the panel it is to be placed on. This type of item has a data type of "String" (chosen from String, Integer or Real), and is of "Presentation Category: Text" and "Presentation Type: Text Display". Additionally, a border width of 3 and a shadow thickness of 2 has been chosen by the developer. The developer has the option to restrict the data to a specified string length and to specify the size (width and height) of the text display area and the location it is to appear on the panel. To adjust the location, the developer merely clicks on the item to "select" it and, holding the mouse button down, drags it to the preferred location. The mechanics of generating the calendar and linking the text will be described below. The initial step is to get the visual component defined. The second type of item required in the day planner is a button. Again, the item must be uniquely named and associated with the parent panel. Buttons are of data type "String, Presentation Category: Selection" and "Presentation Type: Push Button". The item title provides the button label, for example, the title of the QUIT button is "QUIT". This button label can be left or right justified or centered on the button via the "Specify Details" operation. The third type of item, text display and label items are of "String", "Presentation Category: Text" and "Presentation Type: Keyin". The item title labels each with the appropriate hour (0800, 0900, etc.). The "Set Details" operation allows the developer to restrict data entry to 50 characters to prevent the input from overflowing the provided space. In other applications the "Set Constraints" operation can be used to restrict the user's data entry. For example, when entering a new appointment the user would be queried as to the time slot to change. At that point a data entry check would be done to ensure the reply was one of 0800, 0900, 1000...1600, etc.

c. Defining Item Connections

Once the visual layout of the panels and items is specified the developer must define the item connections. The WorkBench mode is changed to "Define Connections". As each object is selected (by clicking on it), a "Connection Specification" window appears allowing the developer to define the disposition of the current panel and to identify the desired subsequent panel. Choices of current panel state after connection to the next panel are: "Preferred" (it remains the active window), "iconic" (the window becomes an icon), "deleted", "invisible" or "no change". Information pertinent to the "next" panel includes "panel name", and "panel state": preferred, iconic, deleted, etc. Additionally, there is an area "TCL command" to enter a TAE+ Command Language string that would be executed upon connection to a subsequent panel.

d. Generating Source Code

When the interface design is complete, the developer generates source code using the automatic generation utility. Languages supported are Ada, C, C++, FORTRAN and high level TAE Command Language(TCL). Each item and the actions resulting from the execution of the item are considered an event. The Code Generator creates event "stubs" that must subsequently be fleshed out by the developer to fully define the desired functionality.

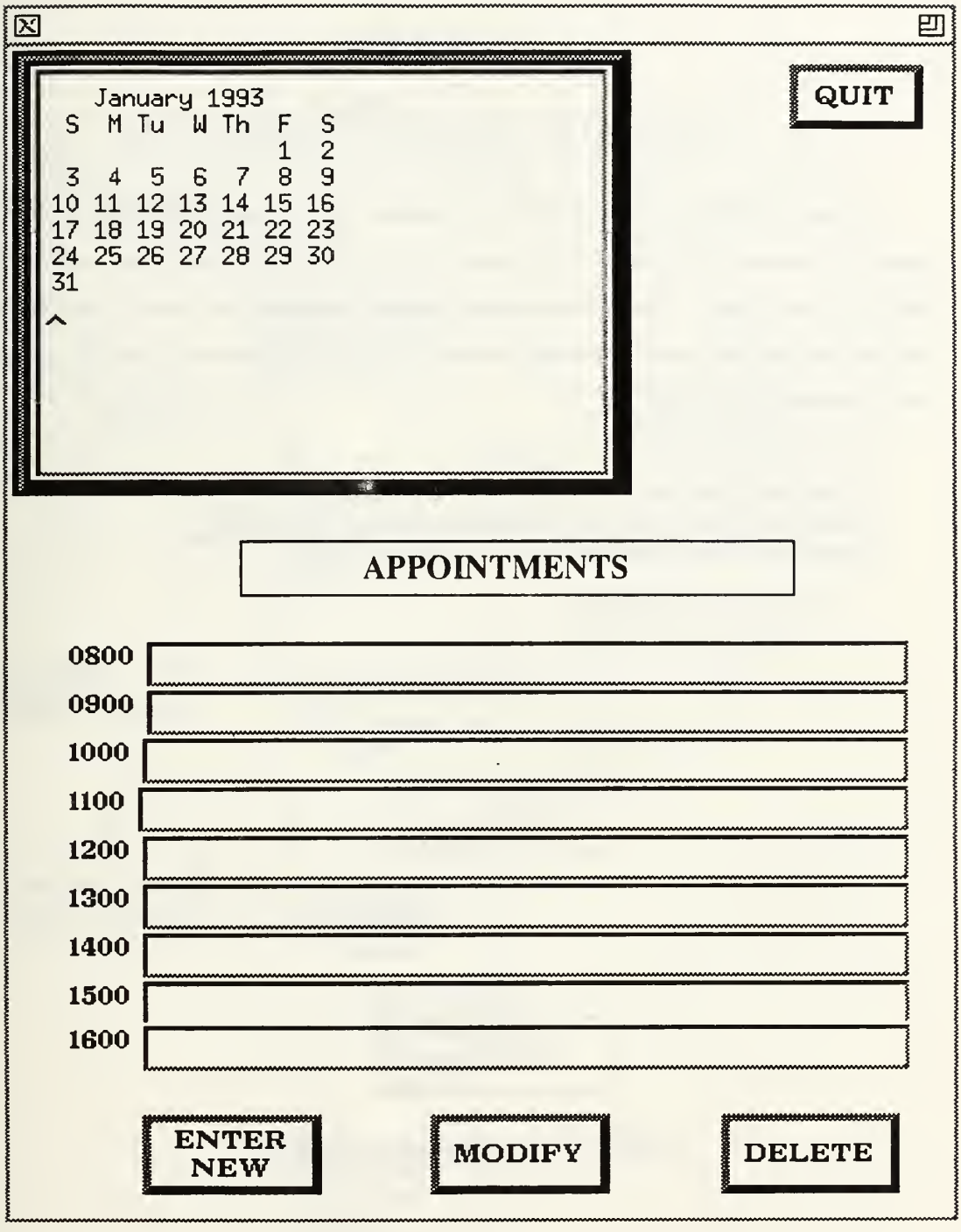


Figure 1. Day Planner Page

APPENDIX B

TAE+ STATE TRANSITION DIAGRAMS (STDs)

Yourdon in [YOURD 89] describes the state-transition diagram notation used in this thesis. Each rectangular *box* represents a state that a system can be in. The *arrows* indicate state changes and each is annotated with a two part label. The *condition* describes the catalyst that causes the state to change and the *action* describes the things that happen as a result of the transition from the former state to the subsequent one. Figure 1 illustrates these concepts.

STD development steps:

1. identify all the possible states OR identify the initial state
2. identify the transactions/conditions (catalyst causing change of state)
3. identify the actions (what happens/results from chg of state)

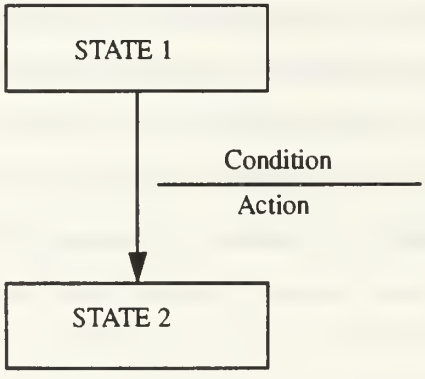
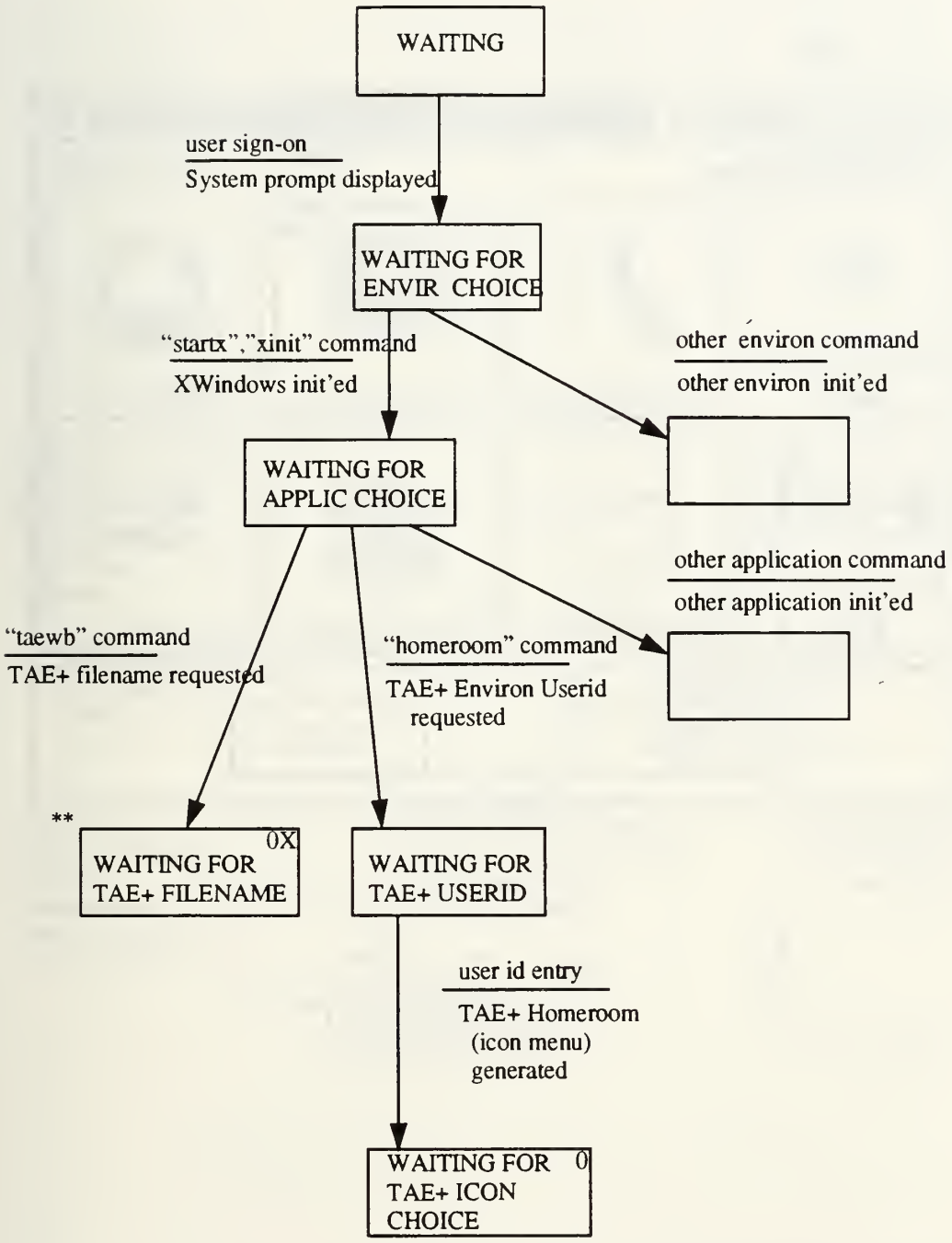
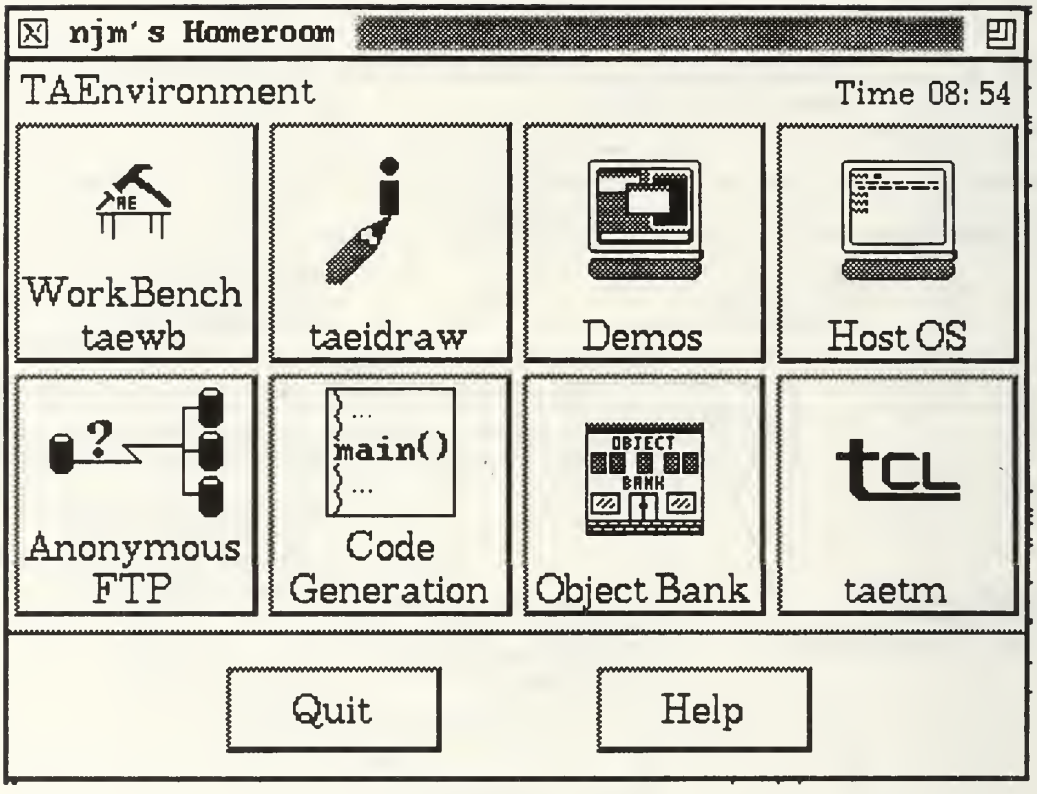


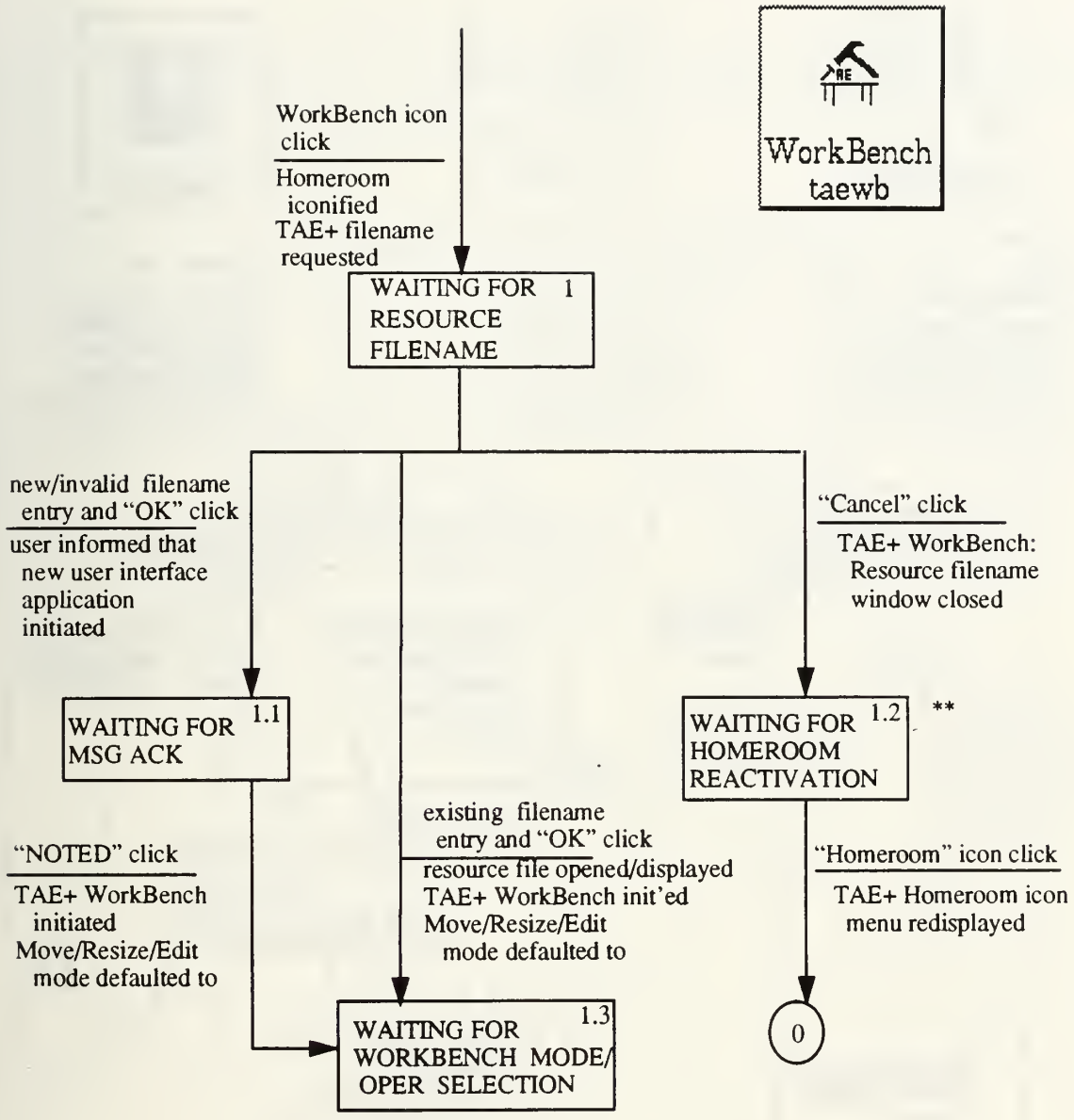
FIGURE 1: STD Constructs [YOURD 89]]

STATE TRANSITION DIAGRAMS FOR TAE+

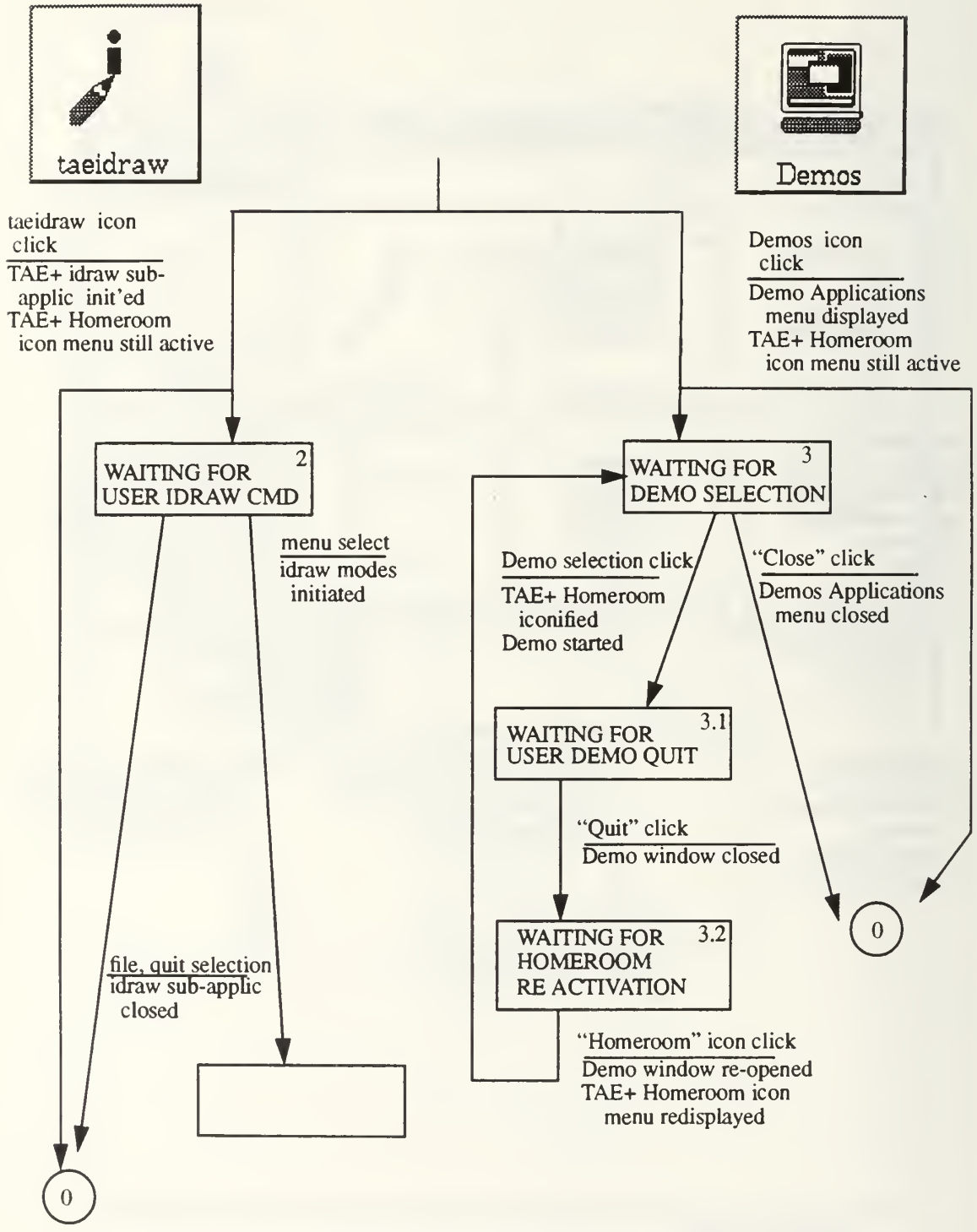


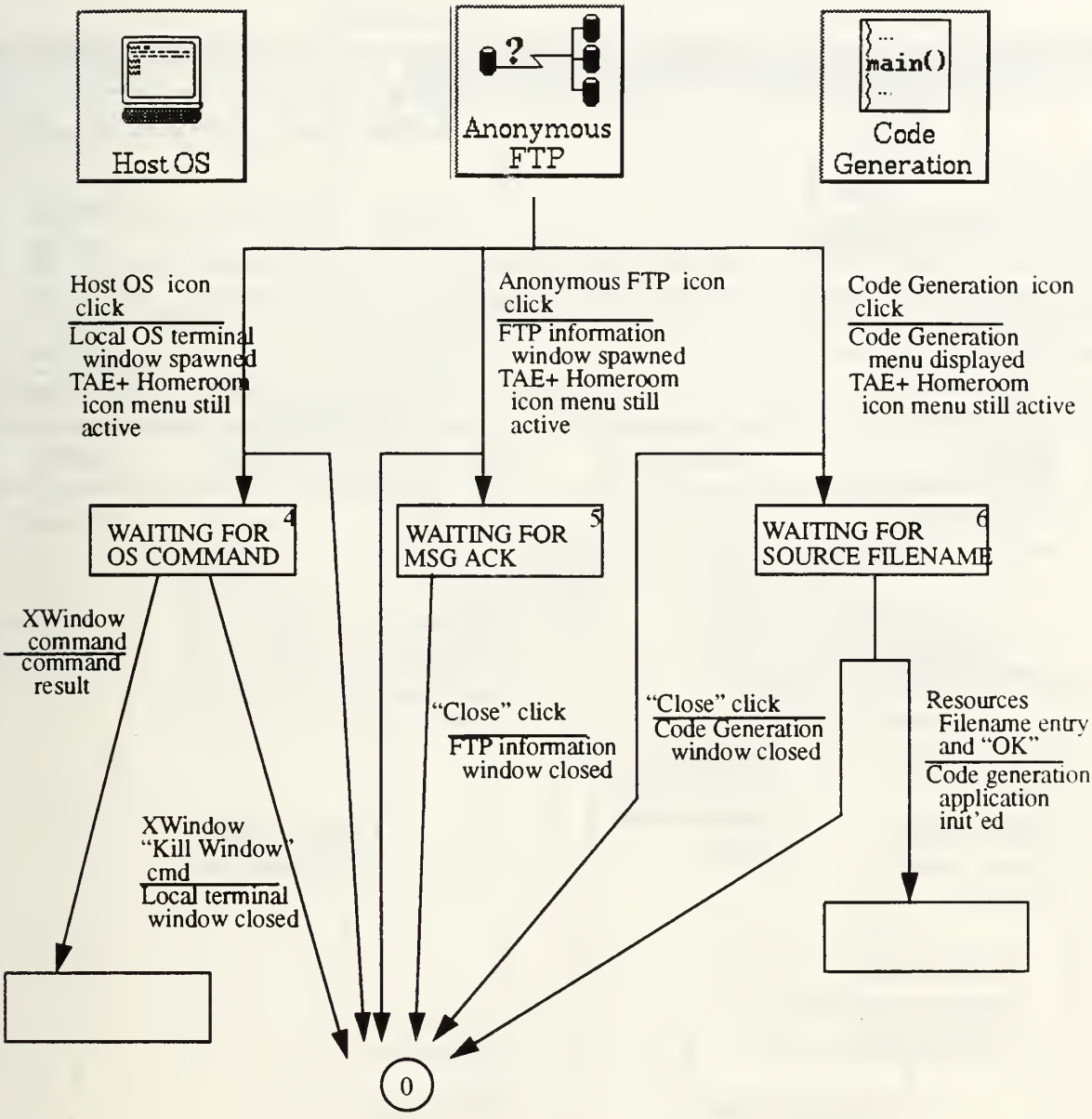
** This is a special case of State 0 in which the TAE+ WorkBench component is activated directly versus via the icon menu. Rather than addressing it separately the minor differences minor differenceswill be noted in depictions of State 0.

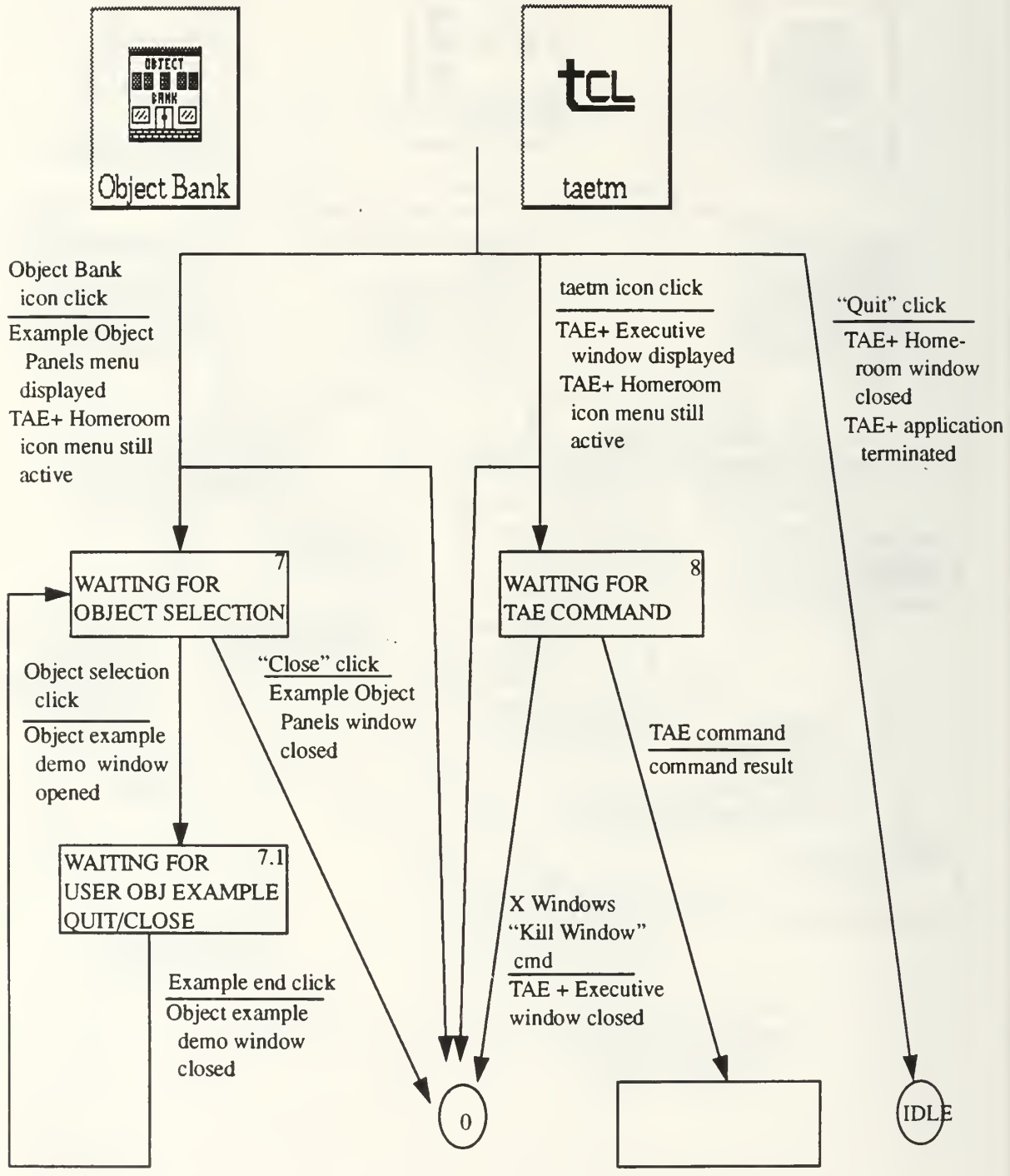

njm's Homeroom
TAEnvironment
Time 08: 54
WorkBench
taewb
taeidraw
Demos
Host OS
Anonymous
FTP
main()
Code
Generation
OBJECT
BANK
Object Bank
tcl
taetm
Quit
Help

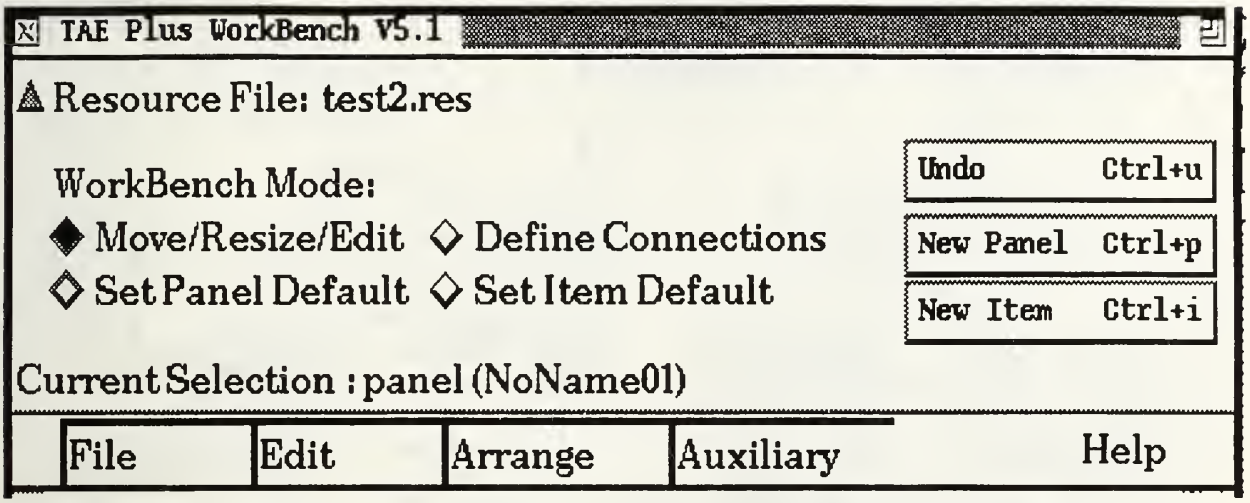


** In the case of only the TAE+ WorkBench being active, the WorkBench window closes and TAE+ terminates.

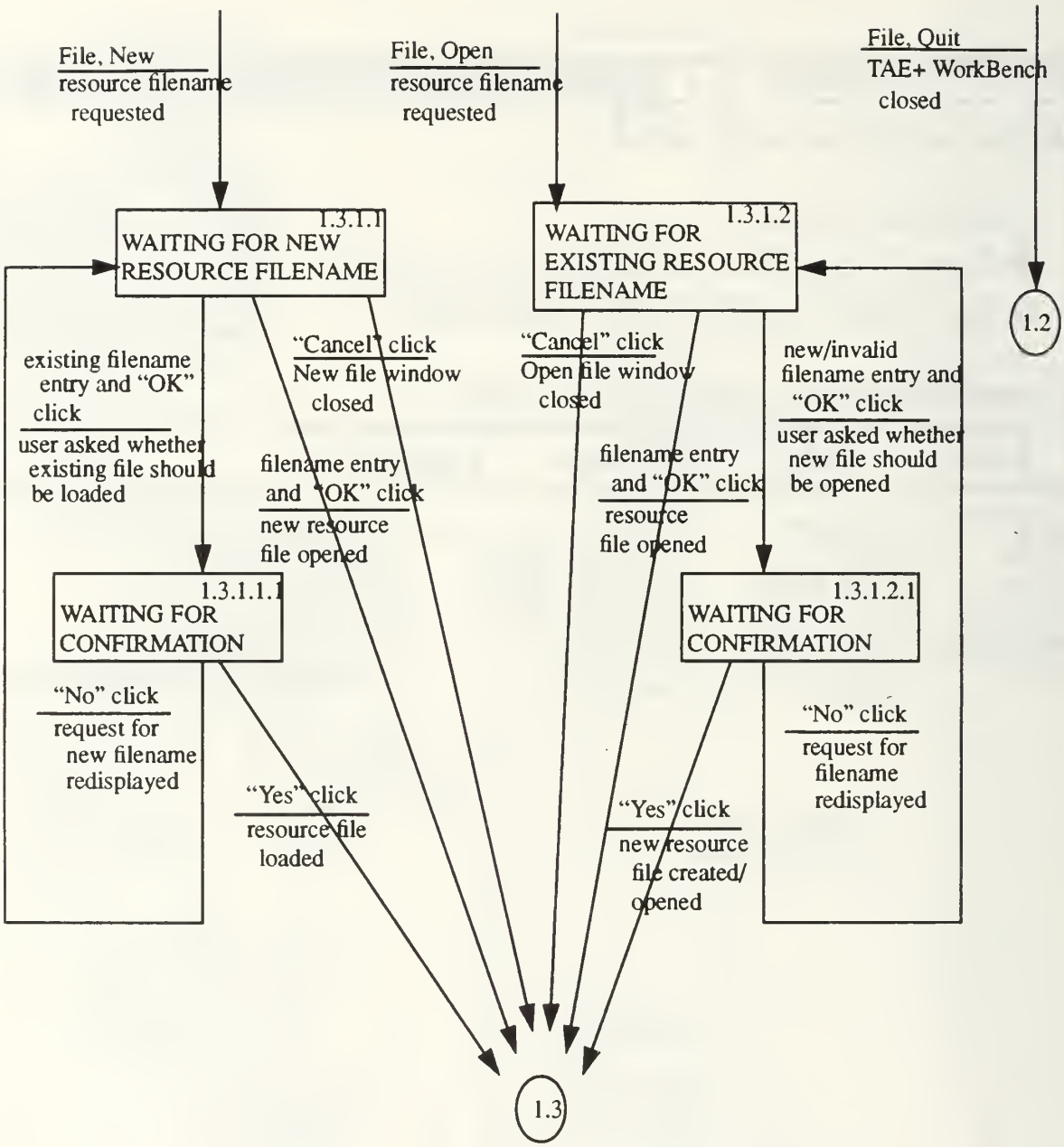

taeidraw
Demos
taeidraw icon
click
TAE+ idraw sub-
applic init'ed
TAE+ Homeroom
icon menu still active
Demos icon
click
Demo Applications
menu displayed
TAE+ Homeroom
icon menu still active
WAITING FOR
USER IDRAW CMD
2
menu select
idraw modes
initiated
WAITING FOR
DEMO SELECTION
3
Demo selection click
TAE+ Homeroom
iconified
Demo started
"Close" click
Demos Applications
menu closed
WAITING FOR
USER DEMO QUIT
3.1
"Quit" click
Demo window closed
WAITING FOR
HOMEROOM
RE ACTIVATION
3.2
"Homeroom" icon click
Demo window re-opened
TAE+ Homeroom icon
menu redisplayed
file, quit selection
idraw sub-applic
closed
0
0

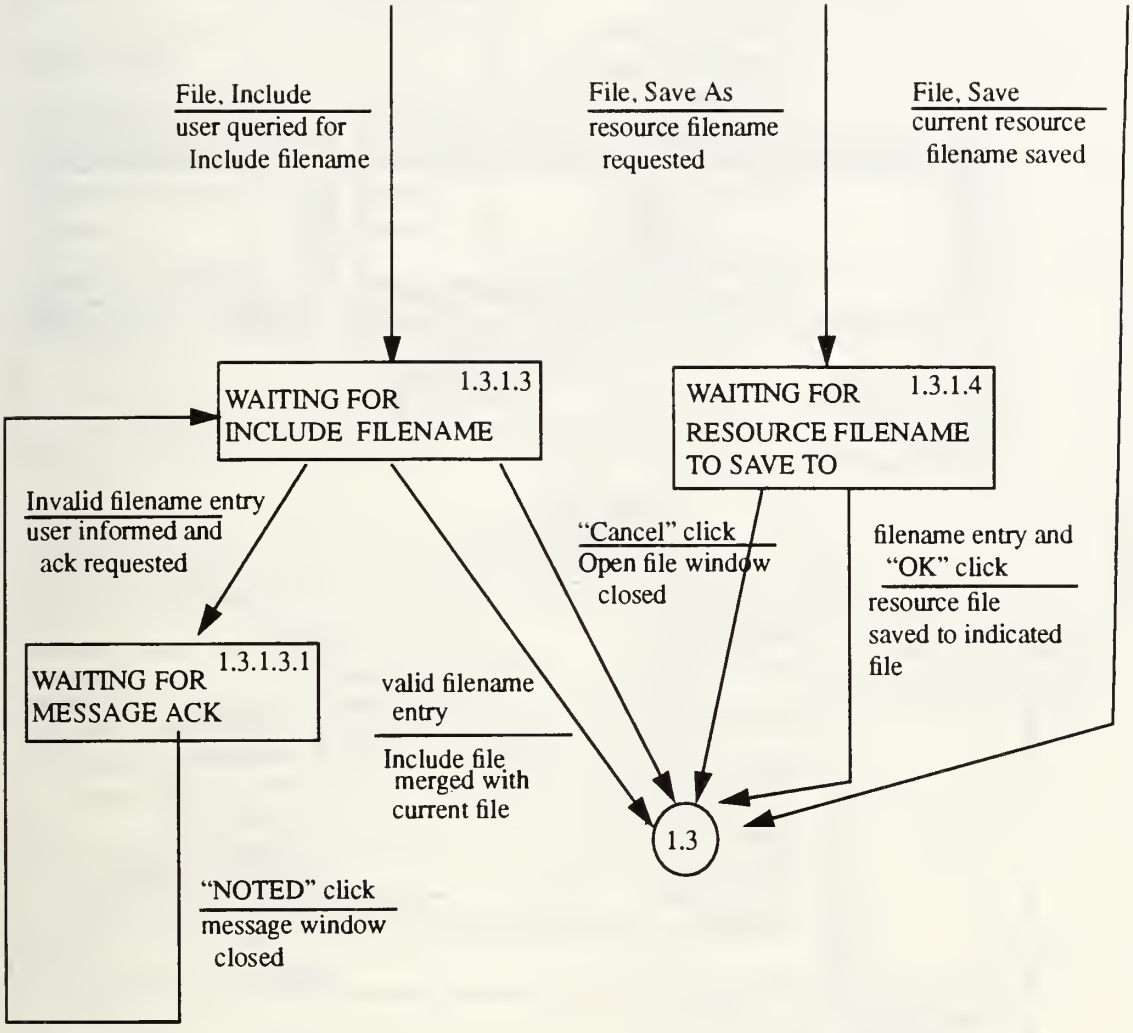

Host OS
Anonymous FTP
Code Generation
main()
Host OS icon click
Local OS terminal window spawned
TAE+ Homeroom icon menu still active
Anonymous FTP icon click
FTP information window spawned
TAE+ Homeroom icon menu still active
Code Generation icon click
Code Generation menu displayed
TAE+ Homeroom icon menu still active
WAITING FOR OS COMMAND
4
WAITING FOR MSG ACK
5
WAITING FOR SOURCE FILENAME
6
XWindow command
command result
XWindow "Kill Window" cmd
Local terminal window closed
"Close" click
FTP information window closed
"Close" click
Code Generation window closed
Resources Filename entry and "OK"
Code generation application init'ed
0

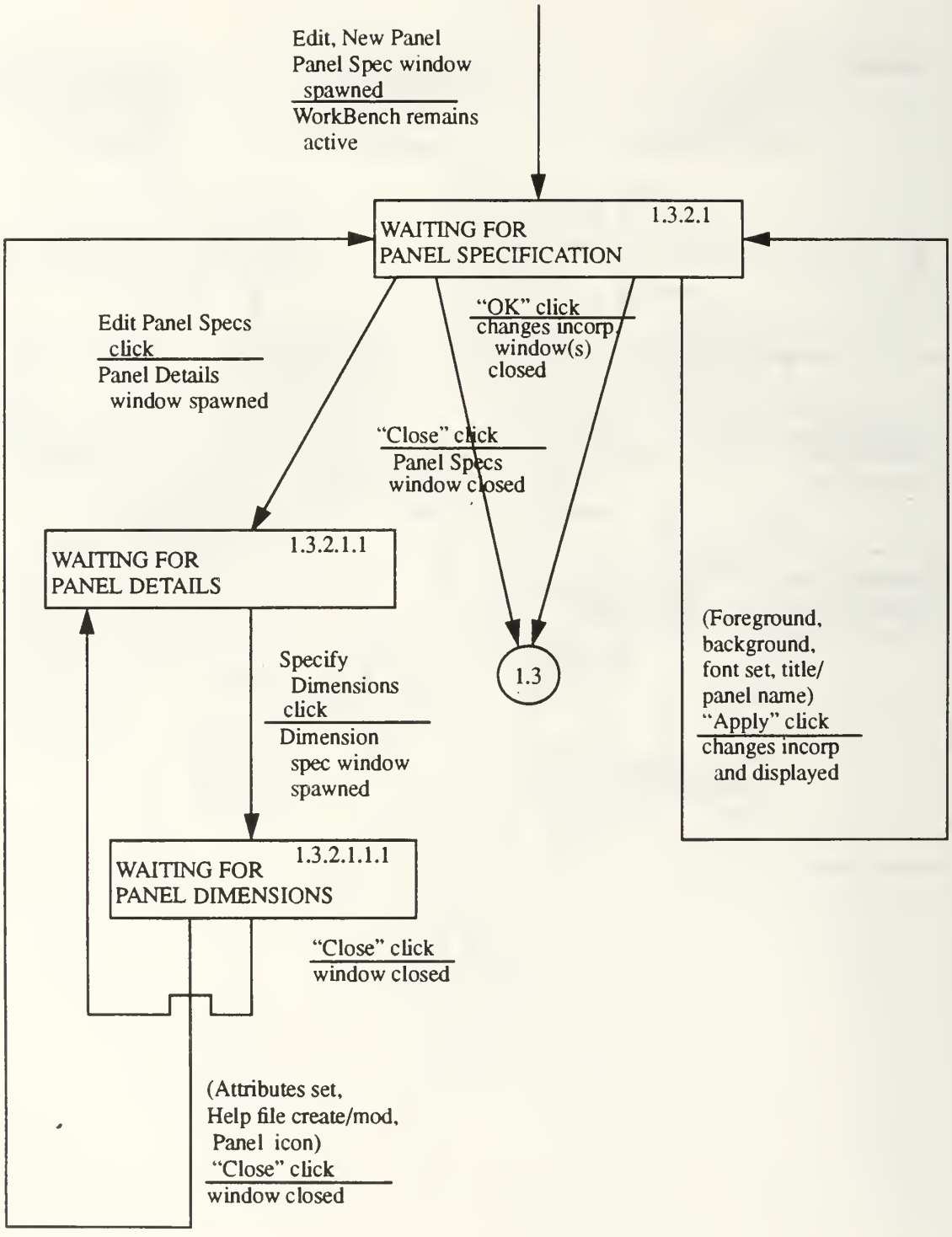

Object Bank
taetm
Object Bank icon click
Example Object Panels menu displayed
TAE+ Homeroom icon menu still active
taetm icon click
TAE+ Executive window displayed
TAE+ Homeroom icon menu still active
"Quit" click
TAE+ Home-room window closed
TAE+ application terminated
7
WAITING FOR OBJECT SELECTION
8
WAITING FOR TAE COMMAND
Object selection click
Object example demo window opened
"Close" click
Example Object Panels window closed
TAE command
command result
7.1
WAITING FOR USER OBJ EXAMPLE QUIT/CLOSE
X Windows "Kill Window" cmd
TAE + Executive window closed
Example end click
Object example demo window closed
0
IDLE

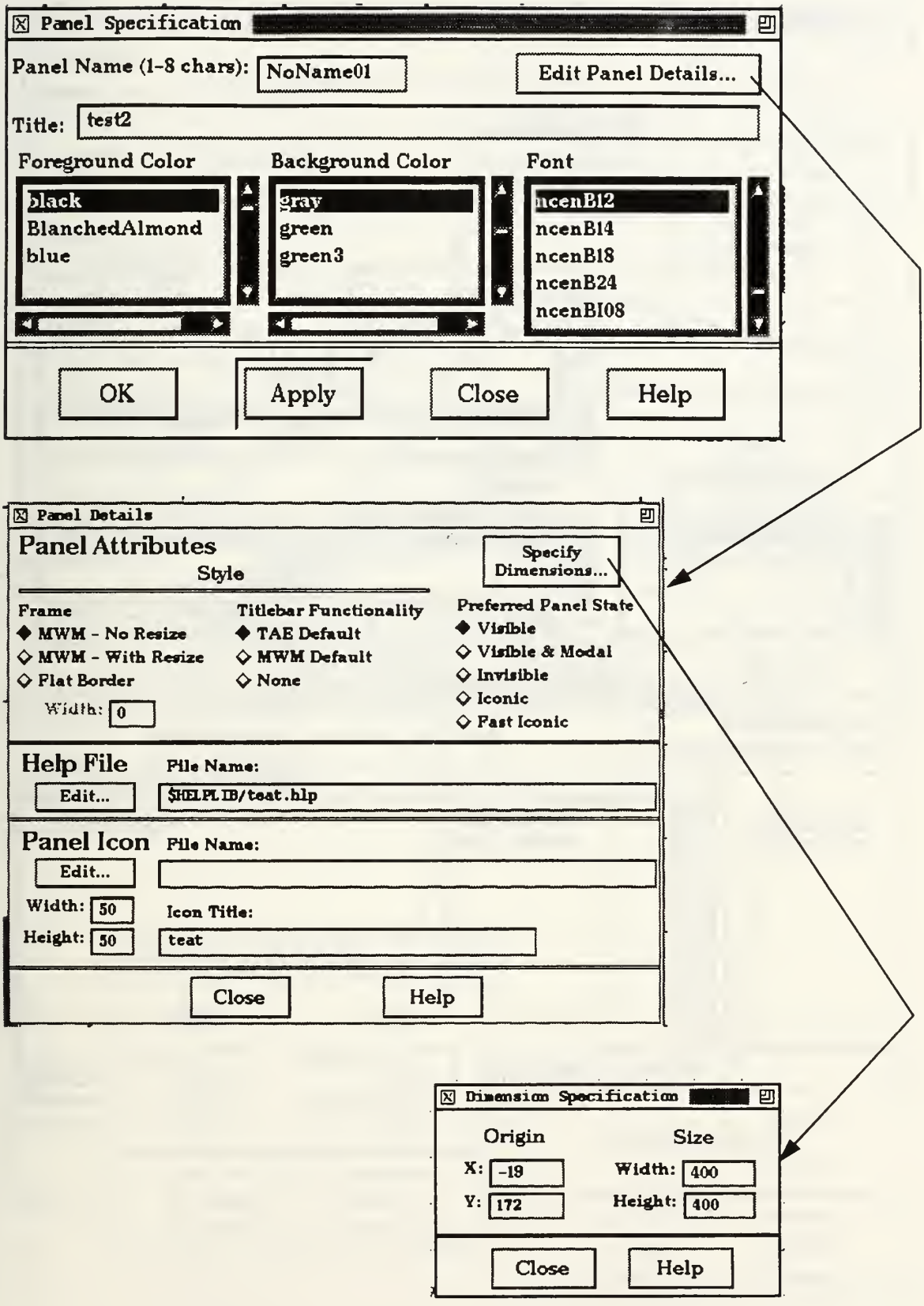


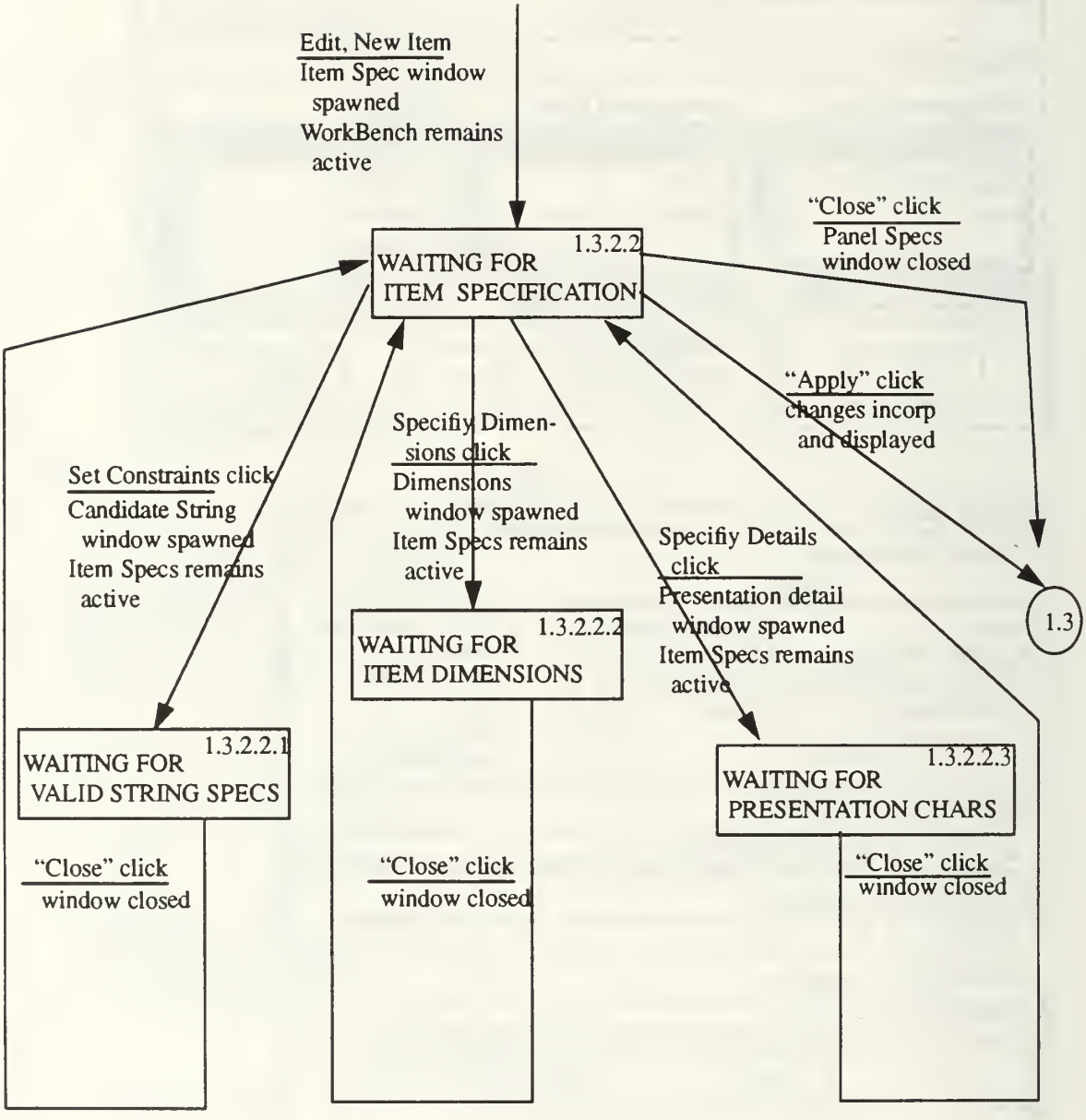
NEW	MODIFY	ALIGN	SPECIFY INIT PANELS
OPEN	UNDO	SPECIFY ALIGN	GENERATE CODE
SAVE	NEW PANEL	TOGGLE GRID	REHEARSE
SAVE AS	NEW ITEM	SNAP TO GRID	ICONIFY WB PANELS
INCLUDE	DUPLICATE	ICONIFY	CREATE TERMINAL
QUIT	SELECT ALL		WORKBENCH PREFERENCES
	DELETE		

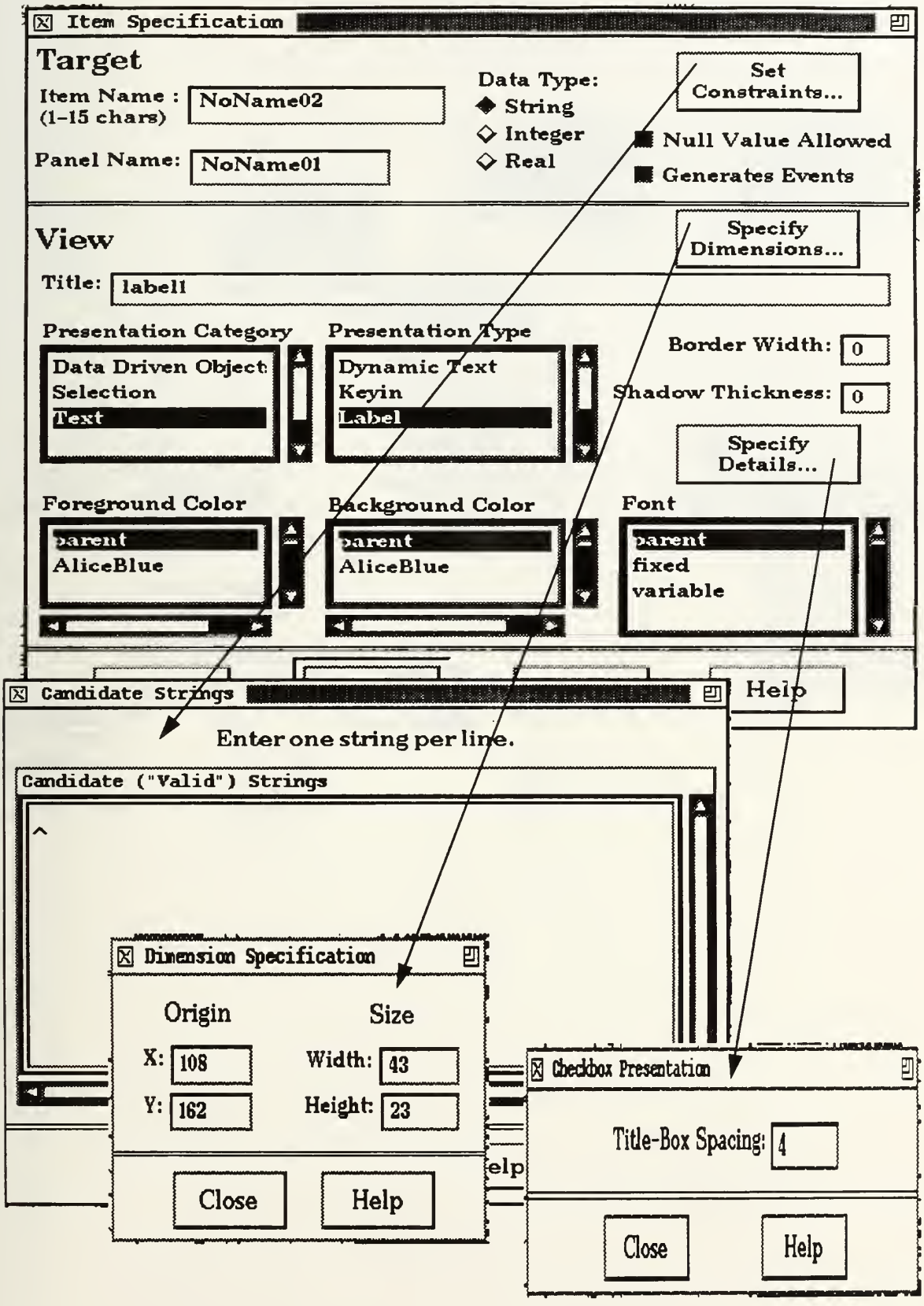

File, New
resource filename
requested
File, Open
resource filename
requested
File, Quit
TAE+ WorkBench
closed
1.3.1.1
WAITING FOR NEW
RESOURCE FILENAME
1.3.1.2
WAITING FOR
EXISTING RESOURCE
FILENAME
1.2
existing filename
entry and "OK"
click
user asked whether
existing file should
be loaded
"Cancel" click
New file window
closed
"Cancel" click
Open file window
closed
new/invalid
filename entry and
"OK" click
user asked whether
new file should
be opened
filename entry
and "OK" click
new resource
file opened
filename entry
and "OK" click
resource
file opened
1.3.1.1.1
WAITING FOR
CONFIRMATION
1.3.1.2.1
WAITING FOR
CONFIRMATION
"No" click
request for
new filename
redisplayed
"Yes" click
resource file
loaded
"Yes" click
new resource
file created/
opened
"No" click
request for
filename
redisplayed
1.3

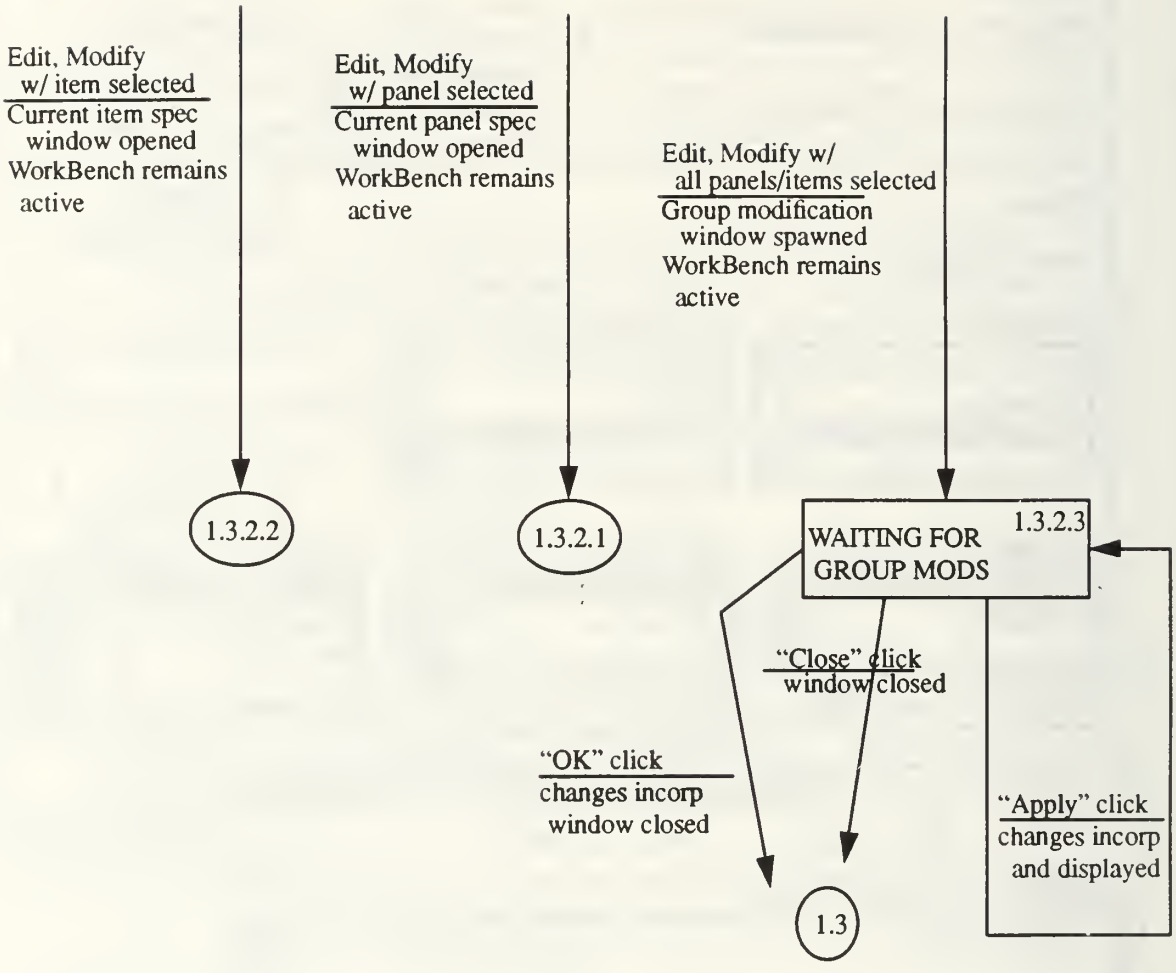

File, Include
user queried for Include filename
File, Save As
resource filename requested
File, Save
current resource filename saved
WAITING FOR INCLUDE FILENAME
1.3.1.3
WAITING FOR RESOURCE FILENAME TO SAVE TO
1.3.1.4
Invalid filename entry
user informed and ack requested
"Cancel" click
Open file window closed
filename entry and "OK" click
resource file saved to indicated file
WAITING FOR MESSAGE ACK
1.3.1.3.1
valid filename entry
Include file merged with current file
1.3
"NOTED" click
message window closed

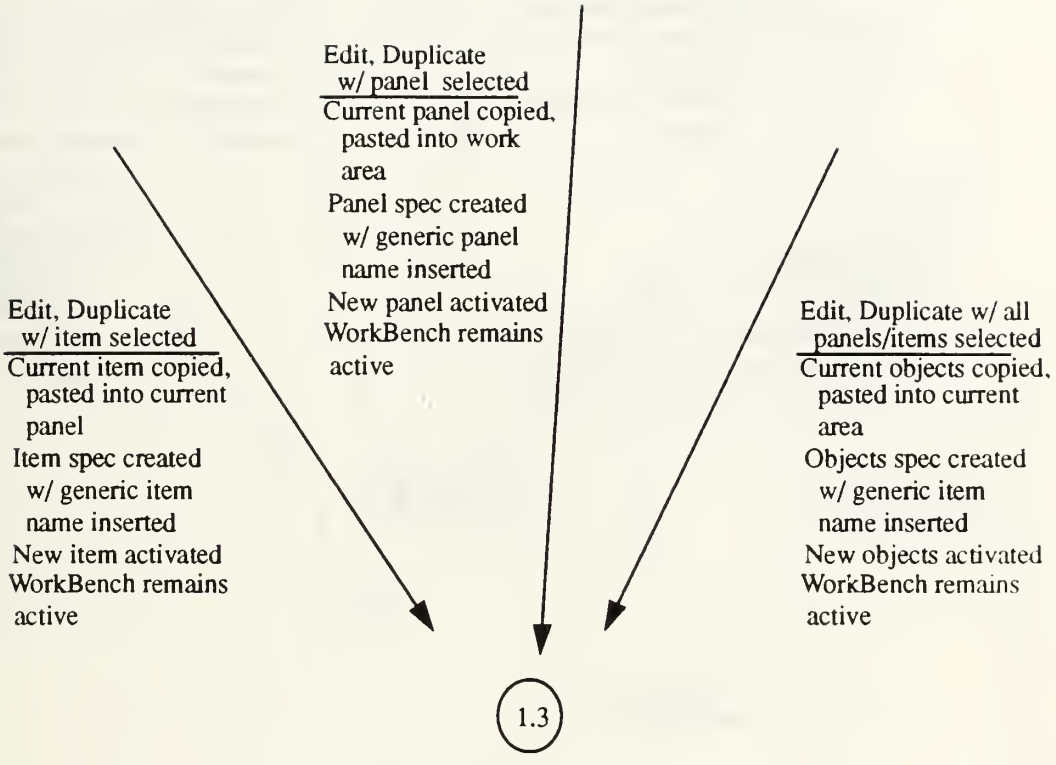

Edit, New Panel
Panel Spec window
spawned
WorkBench remains
active
1.3.2.1
WAITING FOR
PANEL SPECIFICATION
Edit Panel Specs
click
Panel Details
window spawned
"OK" click
changes incorp.
window(s)
closed
"Close" click
Panel Specs
window closed
1.3.2.1.1
WAITING FOR
PANEL DETAILS
1.3
(Foreground,
background,
font set, title/
panel name)
"Apply" click
changes incorp
and displayed
Specify
Dimensions
click
Dimension
spec window
spawned
1.3.2.1.1.1
WAITING FOR
PANEL DIMENSIONS
"Close" click
window closed
(Attributes set,
Help file create/mod,
Panel icon)
"Close" click
window closed

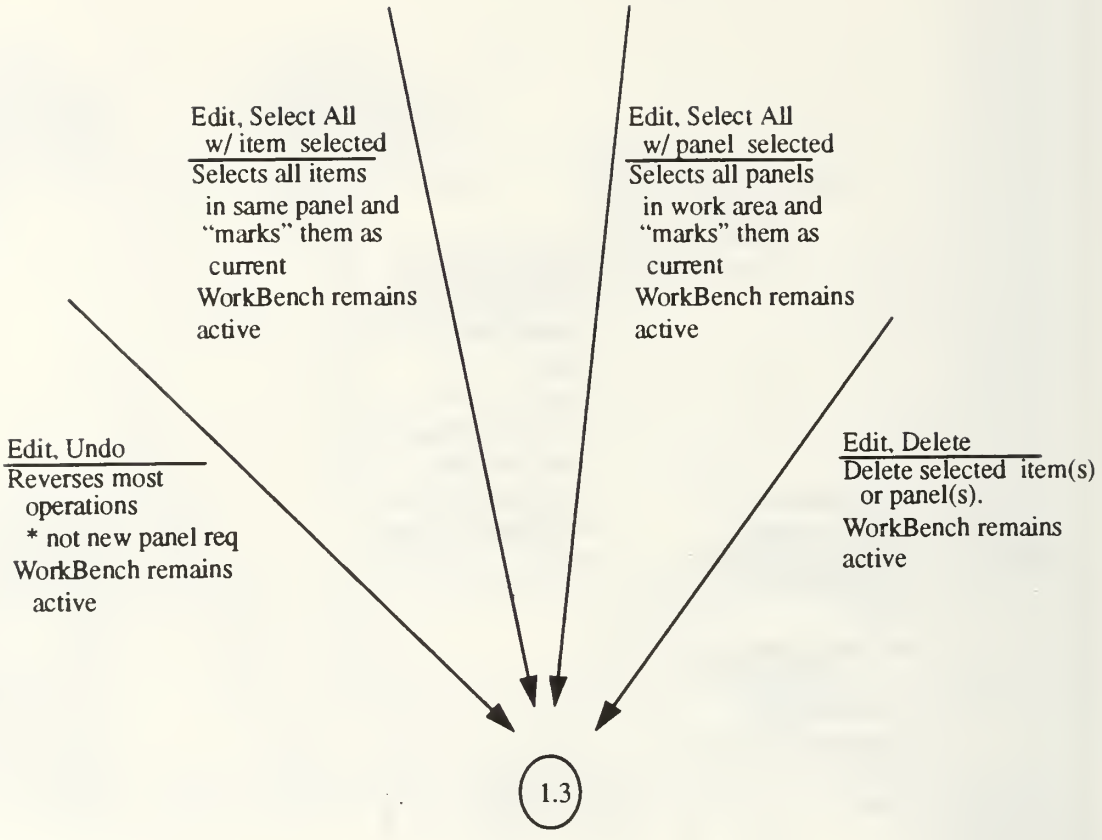

Panel Specification
Panel Name (1-8 chars):
NoName01
Edit Panel Details...
Title:
test2
Foreground Color
Background Color
Font
black
BlanchedAlmond
blue
gray
green
green3
ncenB12
ncenB14
ncenB18
ncenB24
ncenBI08
OK
Apply
Close
Help
Panel Details
Panel Attributes
Style
Specify Dimensions...
Frame
MWM - No Resize
MWM - With Resize
Flat Border
Width:
0
Titlebar Functionality
TAE Default
MWM Default
None
Preferred Panel State
Visible
Visible & Modal
Invisible
Iconic
Past Iconic
Help File
File Name:
Edit...
$HELPLIB/teat.hlp
Panel Icon
File Name:
Edit...
Width:
50
Height:
50
Icon Title:
teat
Close
Help
Dimension Specification
Origin
Size
X:
-19
Y:
172
Width:
400
Height:
400
Close
Help

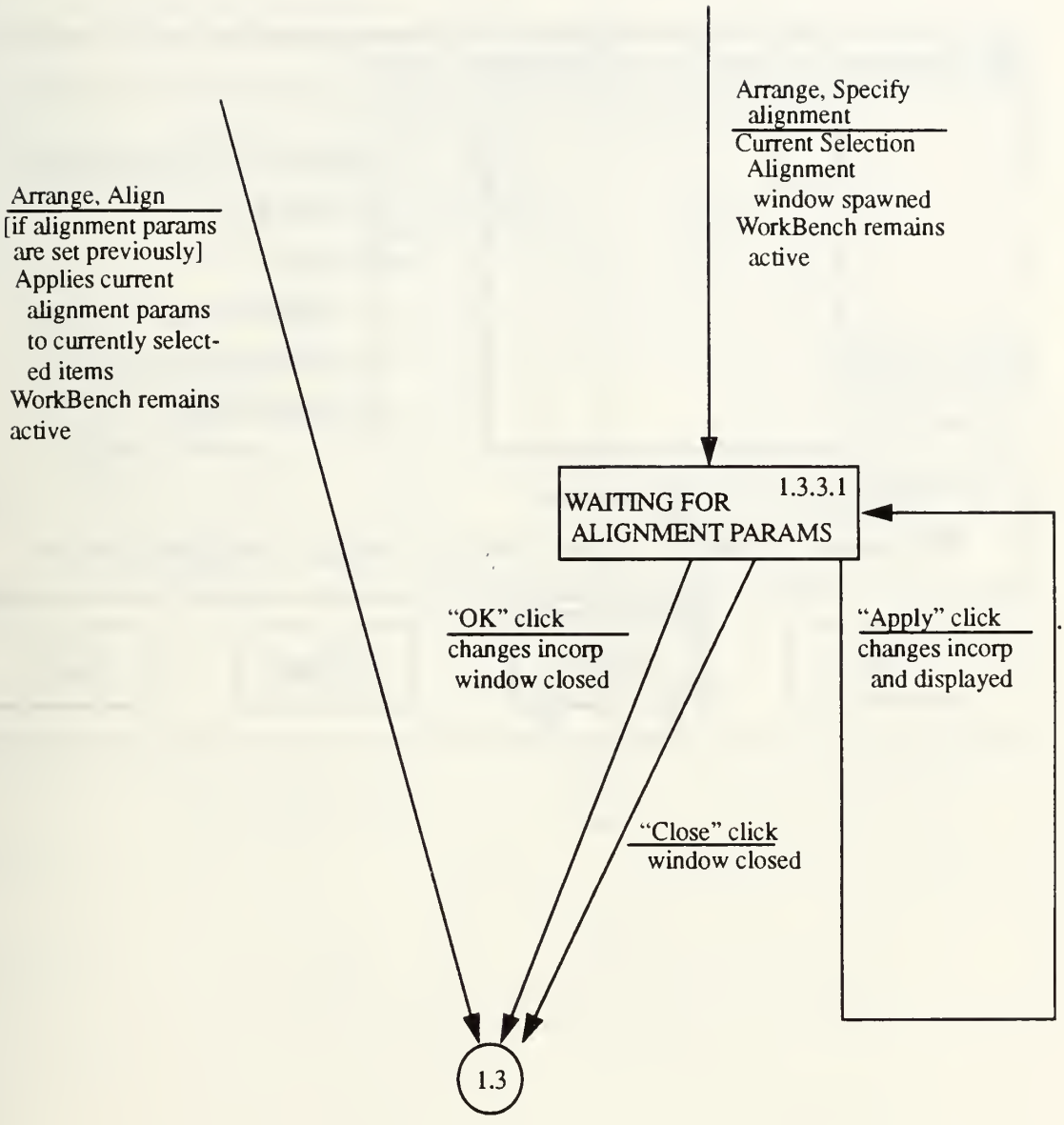

Edit, New Item
Item Spec window spawned
WorkBench remains active
1.3.2.2
WAITING FOR ITEM SPECIFICATION
"Close" click
Panel Specs window closed
"Apply" click
changes incorp and displayed
Set Constraints click
Candidate String window spawned
Item Specs remains active
Specifiy Dimen-sions click
Dimensions window spawned
Item Specs remains active
Specifiy Details click
Presentation detail window spawned
Item Specs remains active
1.3
1.3.2.2.2
WAITING FOR ITEM DIMENSIONS
1.3.2.2.1
WAITING FOR VALID STRING SPECS
1.3.2.2.3
WAITING FOR PRESENTATION CHARS
"Close" click
window closed
"Close" click
window closed
"Close" click
window closed

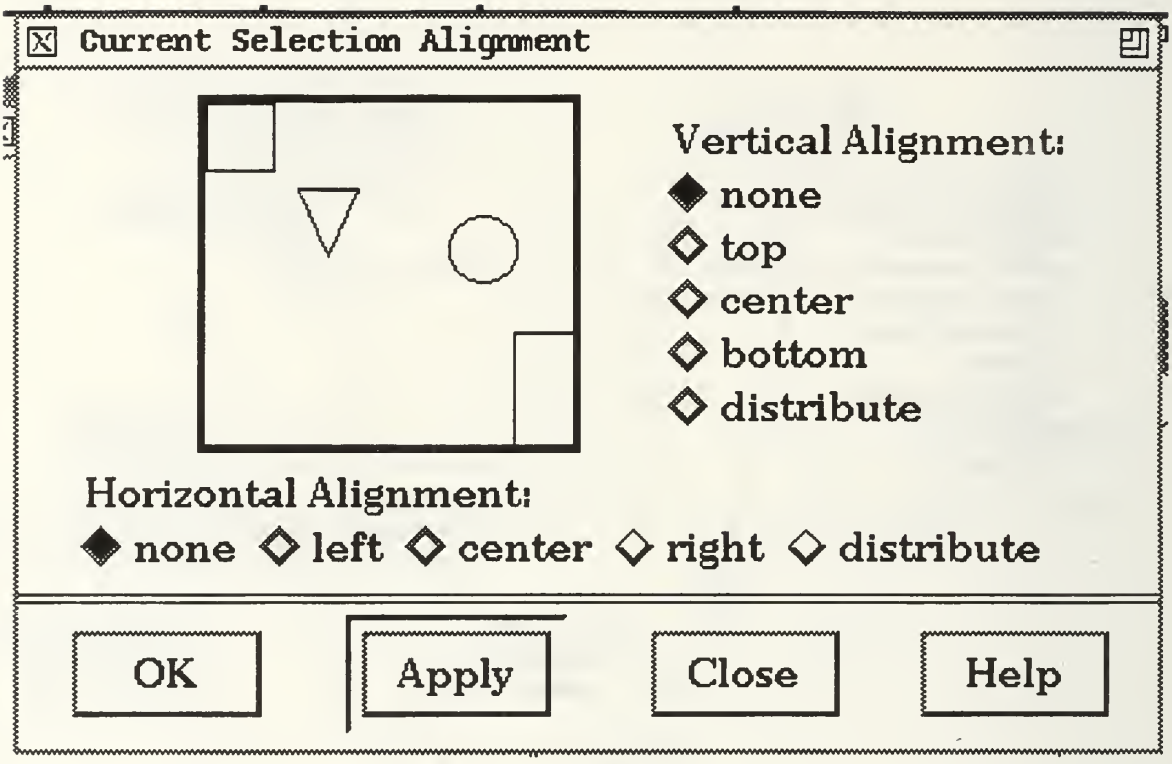

Item Specification
Target
Item Name :
(1-15 chars)
NoName02
Panel Name:
NoName01
Data Type:
String
Integer
Real
Set Constraints...
Null Value Allowed
Generates Events
View
Specify Dimensions...
Title:
label1
Presentation Category
Data Driven Object
Selection
Text
Presentation Type
Dynamic Text
Keyin
Label
Border Width:
0
Shadow Thickness:
0
Specify Details...
Foreground Color
parent
AliceBlue
Background Color
parent
AliceBlue
Font
parent
fixed
variable
Help
Candidate Strings
Enter one string per line.
Candidate ("Valid") Strings
Dimension Specification
Origin
X:
108
Y:
162
Size
Width:
43
Height:
23
Close
Help
Checkbox Presentation
Title-Box Spacing:
4
Close
Help

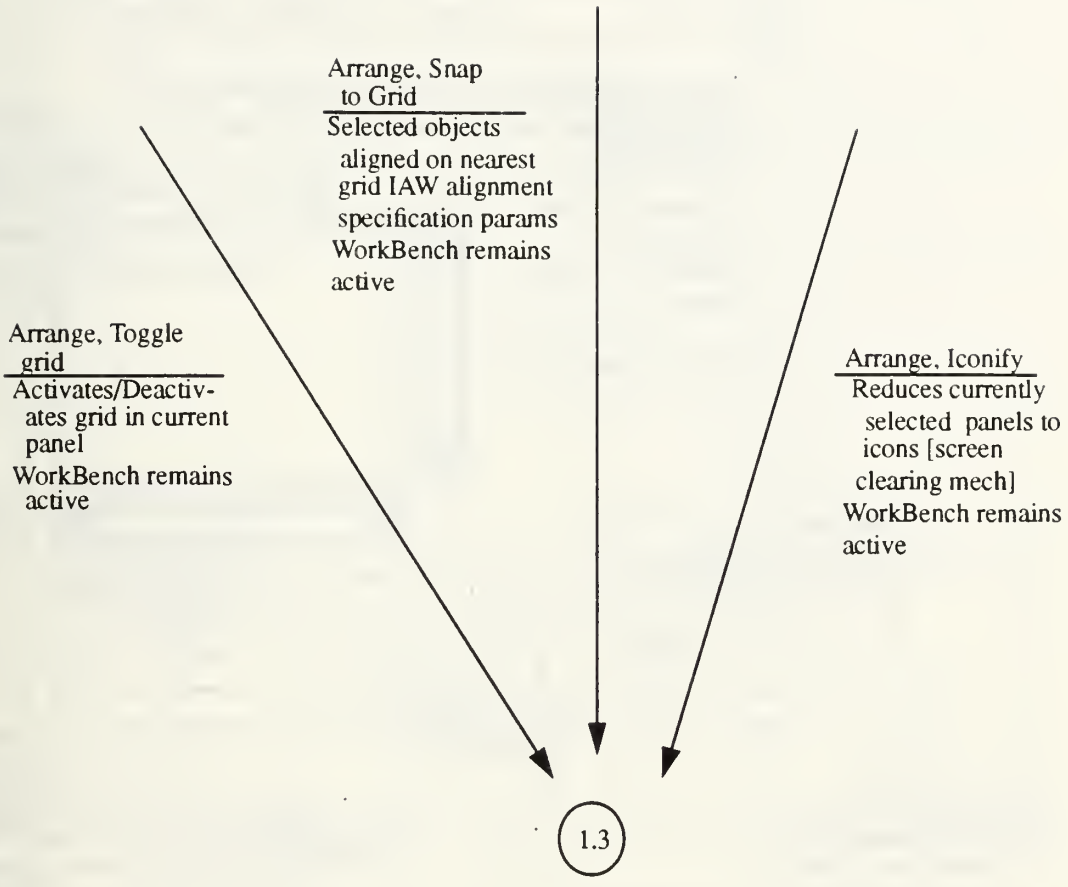

Edit, Modify
w/ item selected
Current item spec
window opened
WorkBench remains
active
Edit, Modify
w/ panel selected
Current panel spec
window opened
WorkBench remains
active
Edit, Modify w/
all panels/items selected
Group modification
window spawned
WorkBench remains
active
1.3.2.2
1.3.2.1
WAITING FOR
GROUP MODS
1.3.2.3
"Close" click
window closed
"OK" click
changes incorp
window closed
"Apply" click
changes incorp
and displayed
1.3

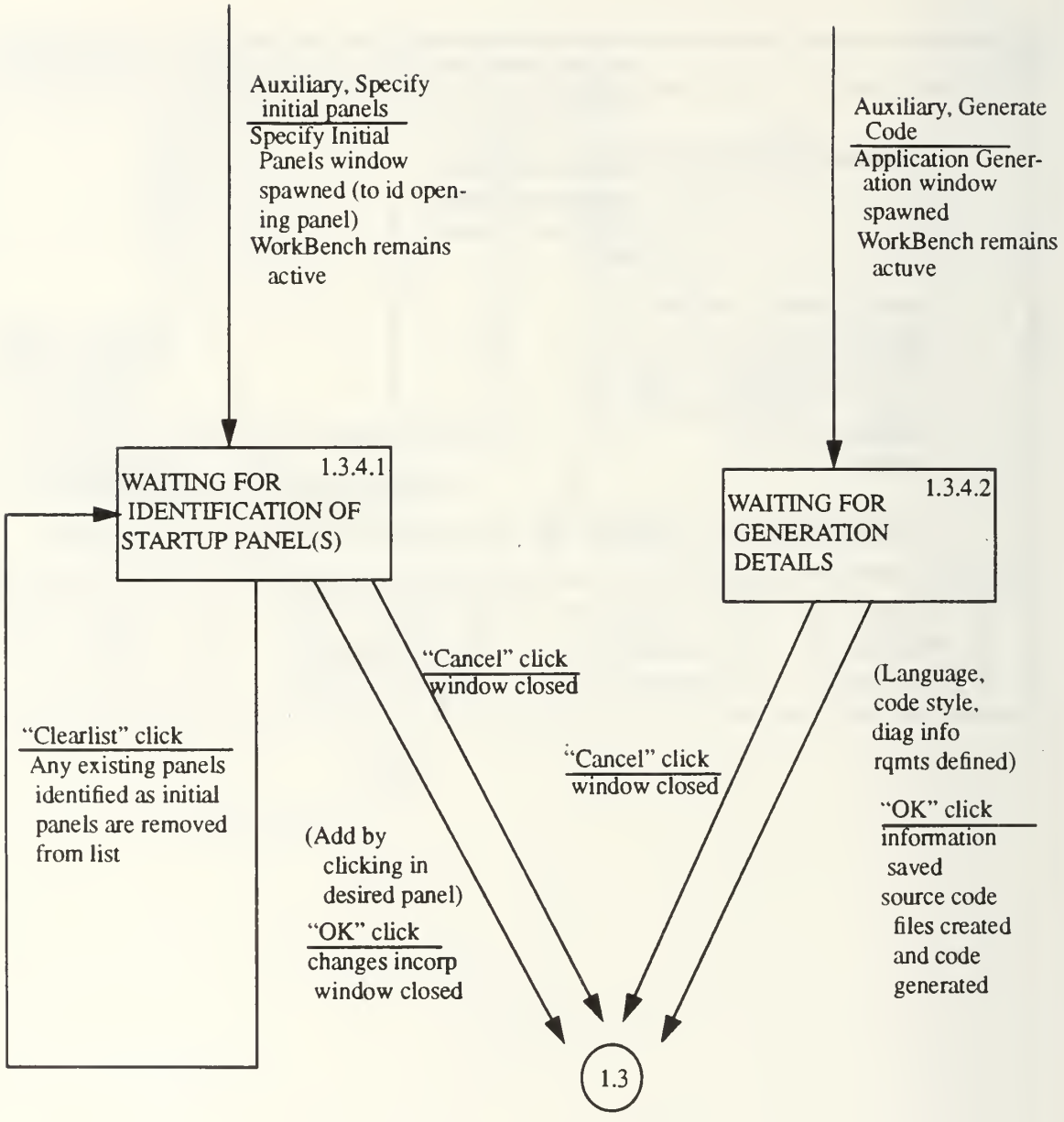

Edit, Duplicate
w/ item selected
Current item copied, pasted into current panel
Item spec created w/ generic item name inserted
New item activated
WorkBench remains active
Edit, Duplicate
w/ panel selected
Current panel copied, pasted into work area
Panel spec created w/ generic panel name inserted
New panel activated
WorkBench remains active
Edit, Duplicate w/ all
panels/items selected
Current objects copied, pasted into current area
Objects spec created w/ generic item name inserted
New objects activated
WorkBench remains active
1.3

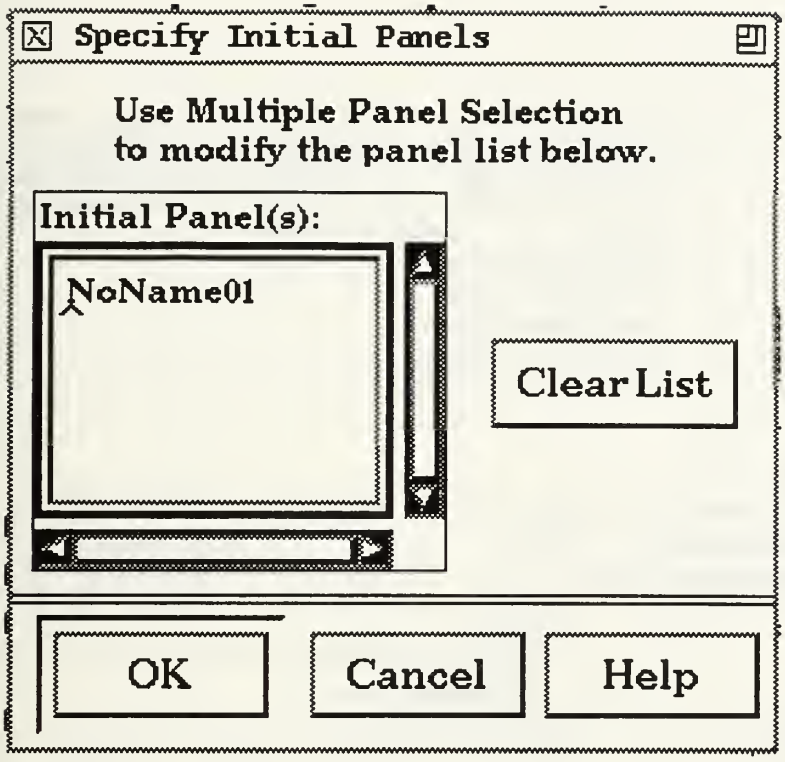

Edit, Select All
w/ item selected
Selects all items
in same panel and
"marks" them as
current
WorkBench remains
active
Edit, Select All
w/ panel selected
Selects all panels
in work area and
"marks" them as
current
WorkBench remains
active
Edit, Undo
Reverses most
operations
* not new panel req
WorkBench remains
active
Edit, Delete
Delete selected item(s)
or panel(s).
WorkBench remains
active
1.3

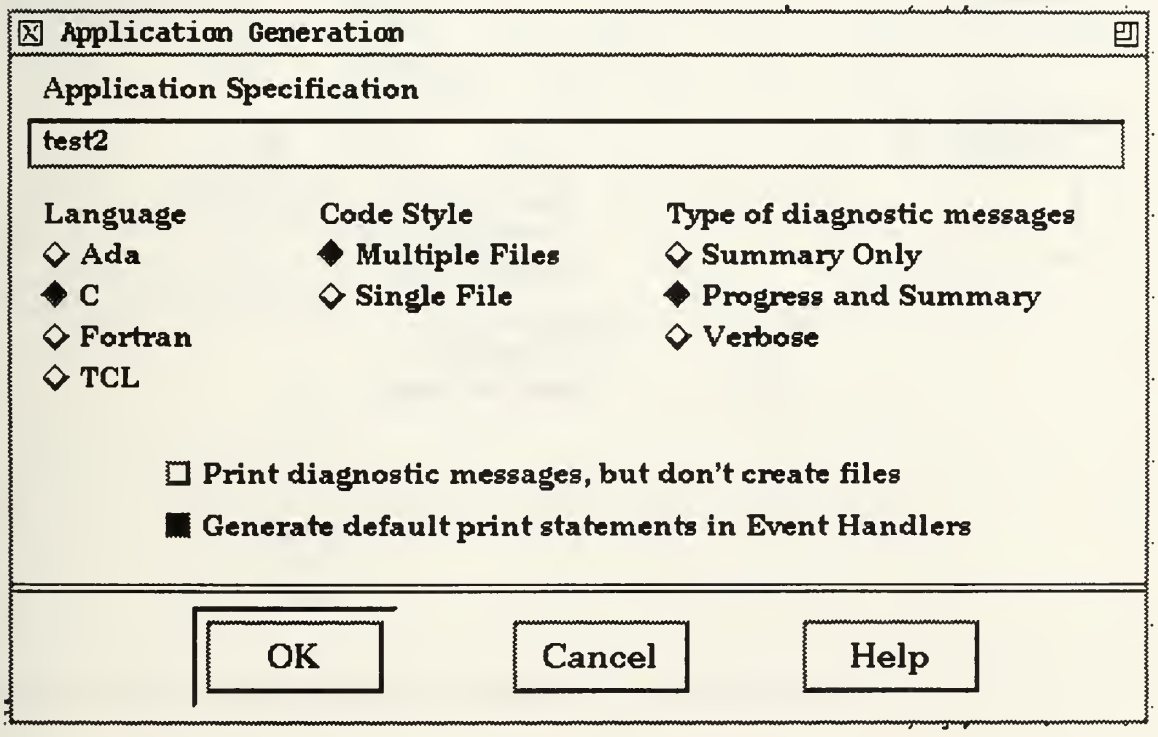

Arrange, Specify alignment
Current Selection Alignment window spawned
WorkBench remains active
Arrange, Align
[if alignment params are set previously]
Applies current alignment params to currently selected items
WorkBench remains active
WAITING FOR ALIGNMENT PARAMS
1.3.3.1
"OK" click
changes incorp
window closed
"Apply" click
changes incorp
and displayed
"Close" click
window closed
1.3

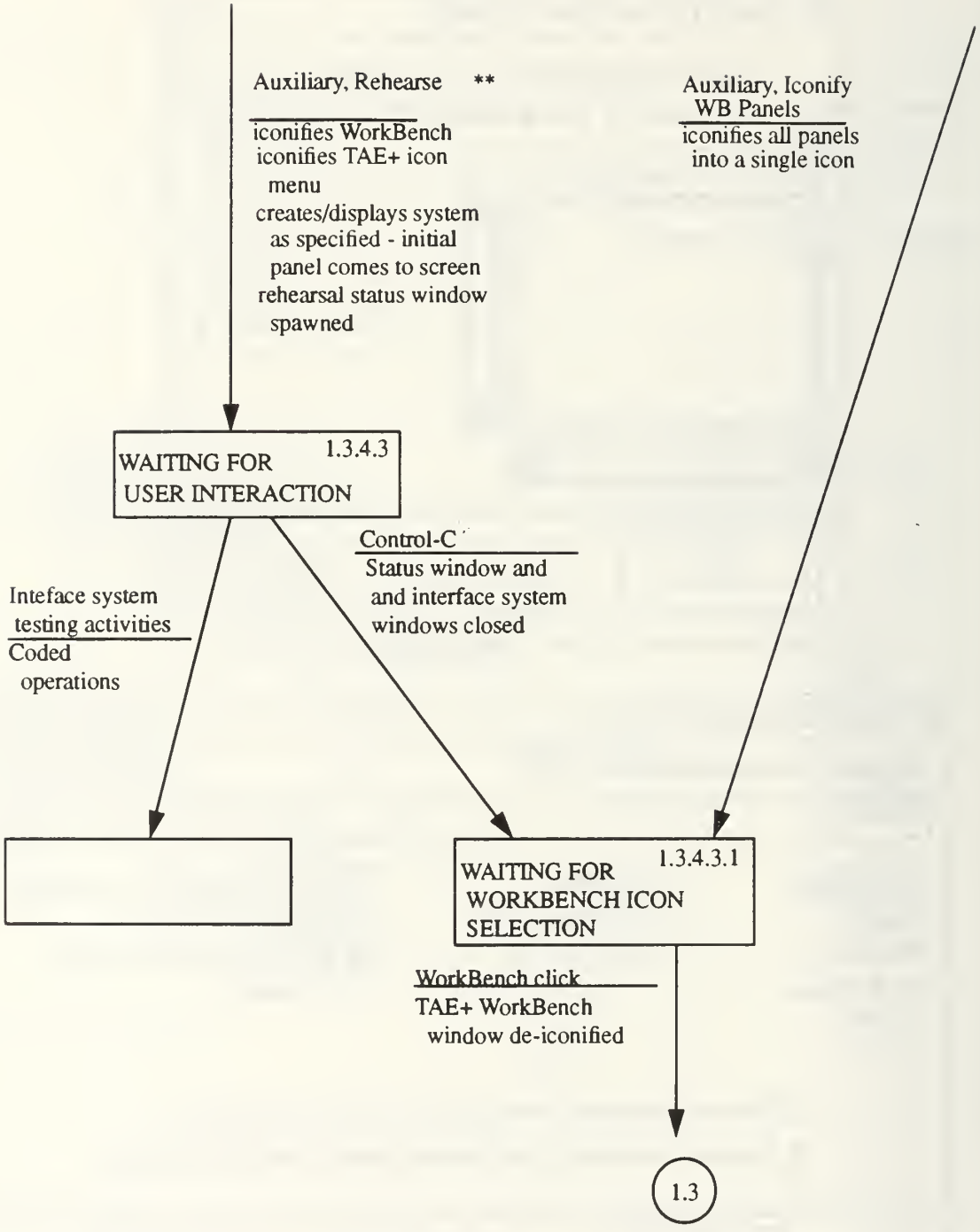

Current Selection Alignment
Vertical Alignment:
none
top
center
bottom
distribute
Horizontal Alignment:
none
left
center
right
distribute
OK
Apply
Close
Help


Arrange, Snap to Grid
Selected objects aligned on nearest grid IAW alignment specification params
WorkBench remains active
Arrange, Toggle grid
Activates/Deactivates grid in current panel
WorkBench remains active
Arrange, Iconify
Reduces currently selected panels to icons [screen clearing mech]
WorkBench remains active
1.3

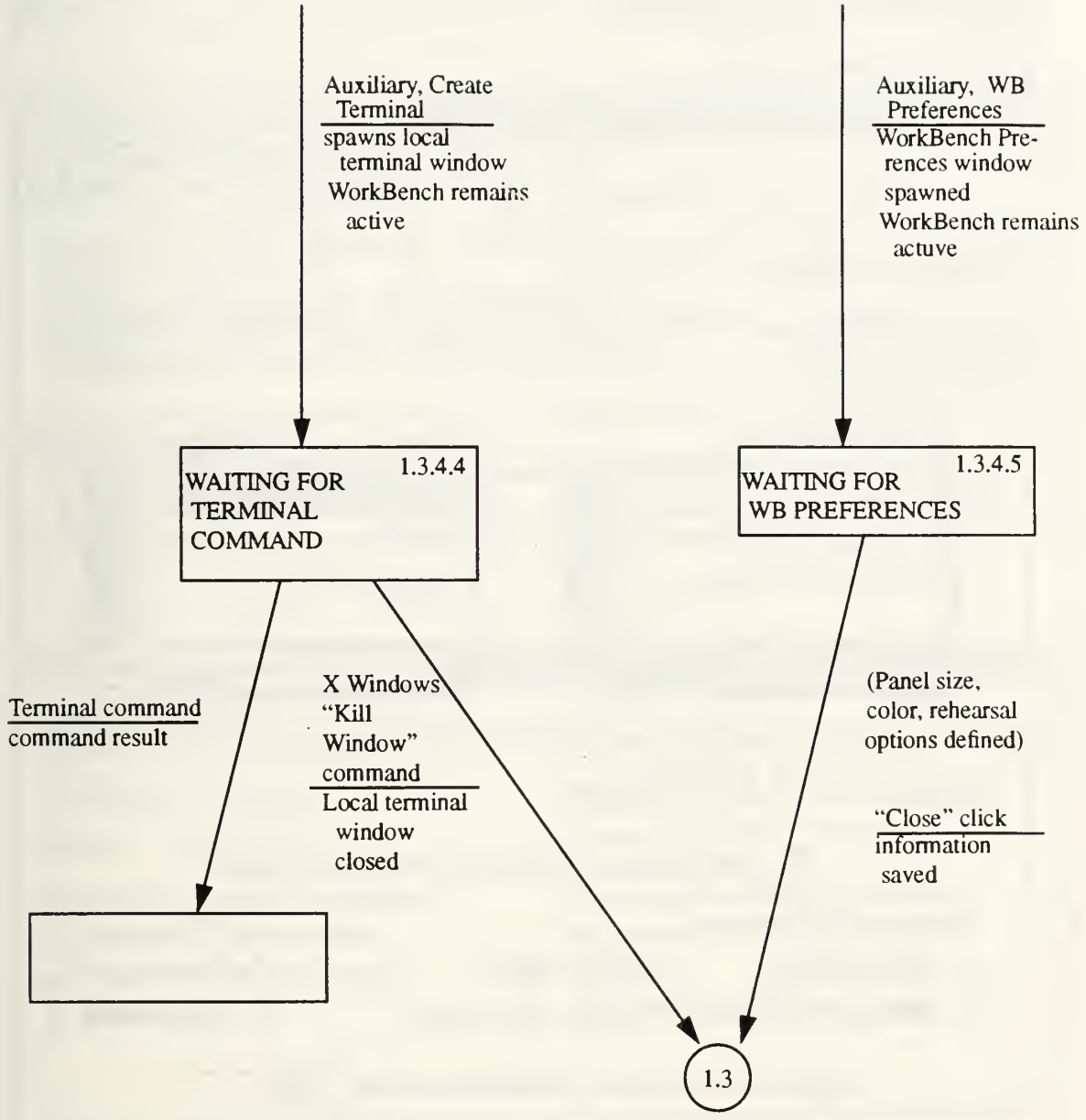

Auxiliary, Specify initial panels
Specify Initial Panels window spawned (to id opening panel)
WorkBench remains active
Auxiliary, Generate Code
Application Generation window spawned
WorkBench remains actuve
1.3.4.1
WAITING FOR IDENTIFICATION OF STARTUP PANEL(S)
1.3.4.2
WAITING FOR GENERATION DETAILS
"Cancel" click
window closed
"Cancel" click
window closed
"Clearlist" click
Any existing panels identified as initial panels are removed from list
(Add by clicking in desired panel)
"OK" click
changes incorp
window closed
(Language, code style, diag info rqmts defined)
"OK" click
information saved
source code files created and code generated
1.3

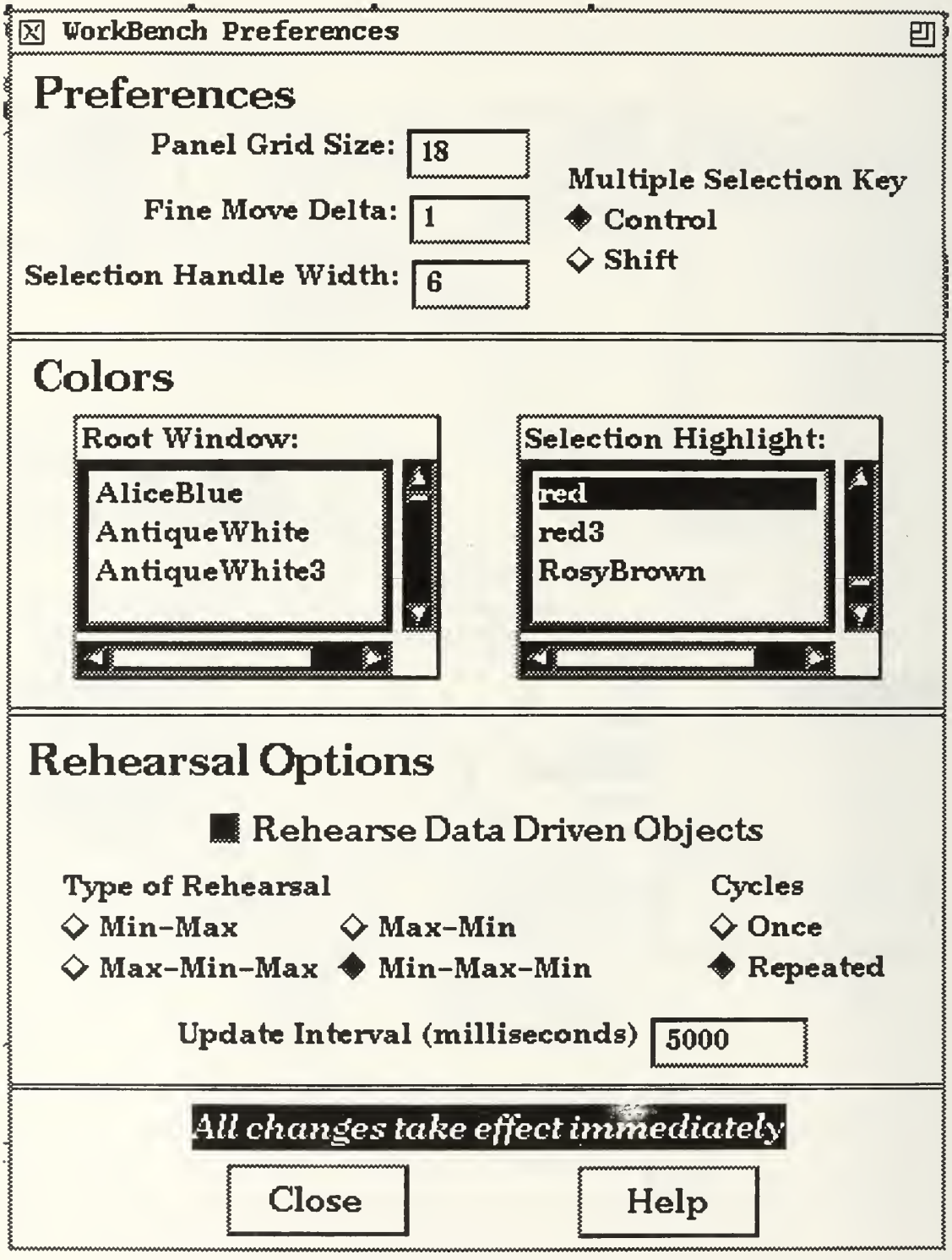

Specify Initial Panels
Use Multiple Panel Selection
to modify the panel list below.
Initial Panel(s):
NoName01
Clear List
OK
Cancel
Help

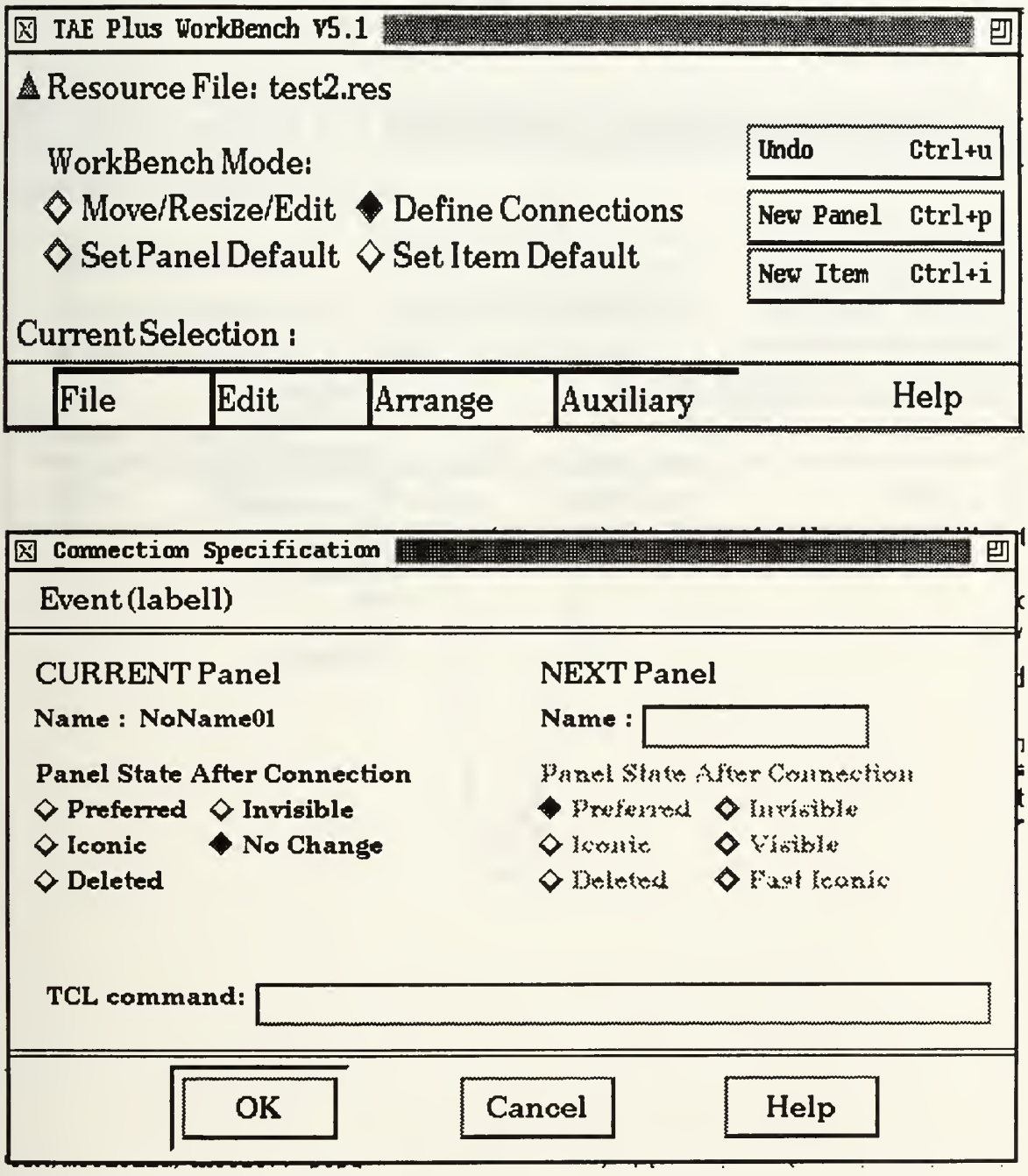

Application Generation
Application Specification
test2
Language
Ada
C
Fortran
TCL
Code Style
Multiple Files
Single File
Type of diagnostic messages
Summary Only
Progress and Summary
Verbose
Print diagnostic messages, but don't create files
Generate default print statements in Event Handlers
OK
Cancel
Help

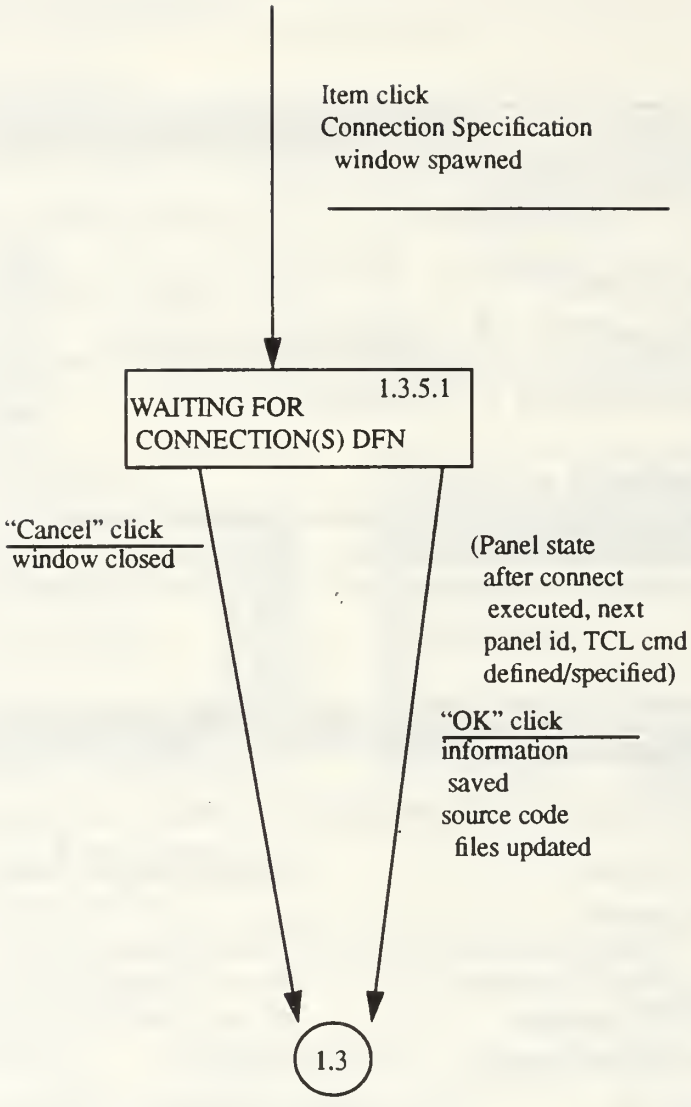


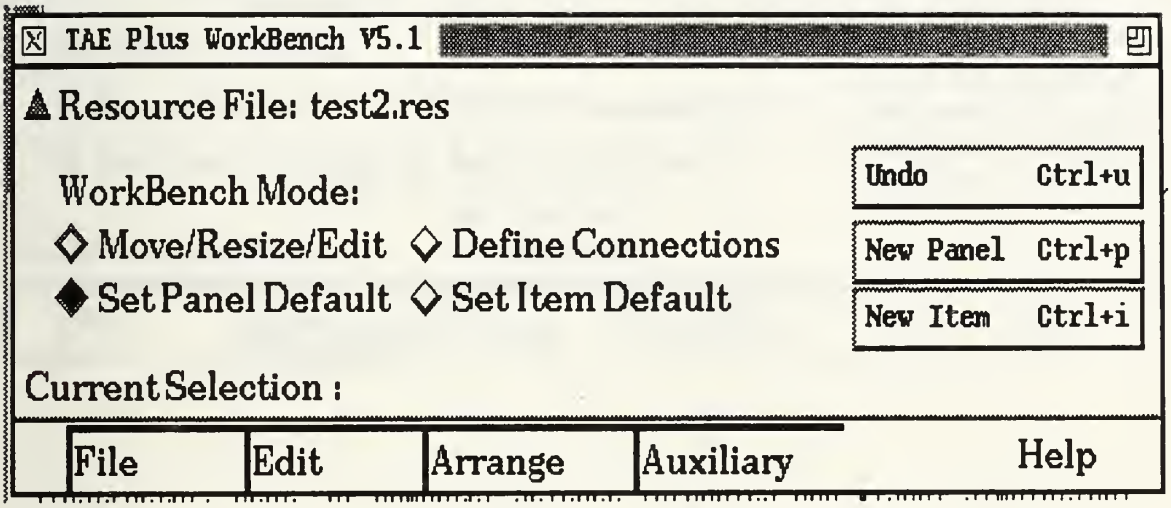
** In case of only the TAE+ WorkBench being active, only the WorkBench window is iconified

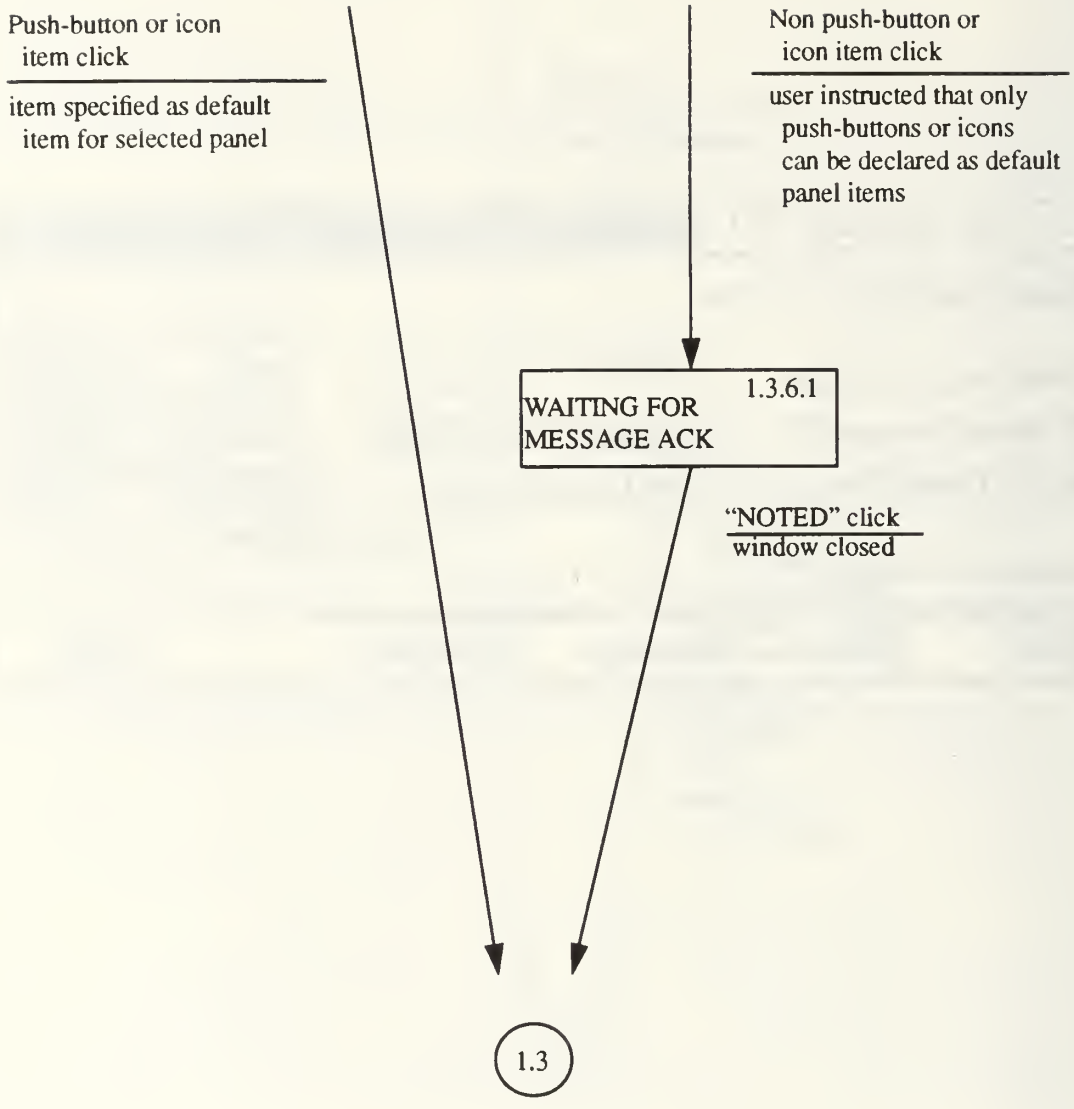

Auxiliary, Create Terminal
spawns local terminal window
WorkBench remains active
Auxiliary, WB Preferences
WorkBench Preferences window spawned
WorkBench remains actuve
WAITING FOR TERMINAL COMMAND
1.3.4.4
WAITING FOR WB PREFERENCES
1.3.4.5
Terminal command
command result
X Windows "Kill Window" command
Local terminal window closed
(Panel size, color, rehearsal options defined)
"Close" click
information saved
1.3

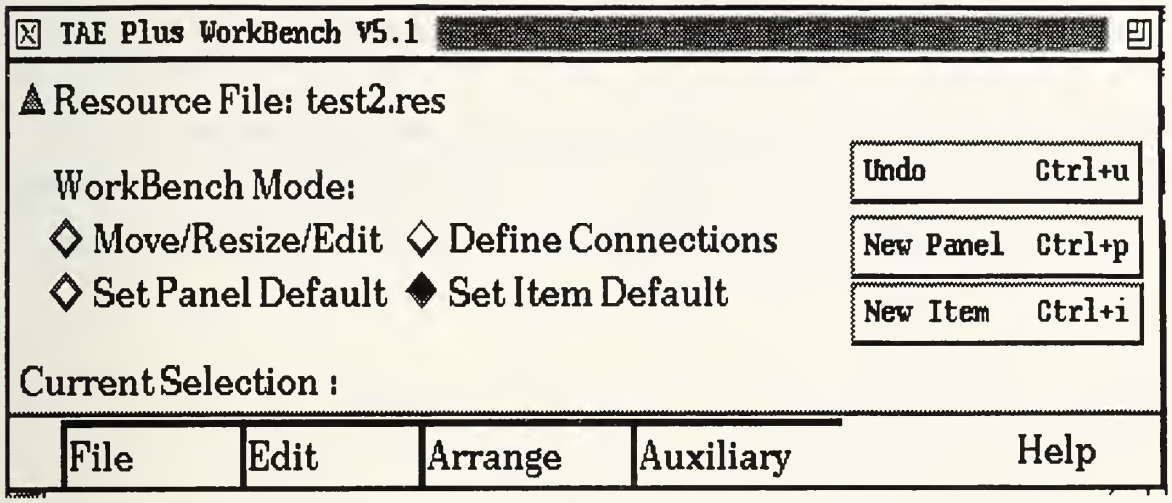

WorkBench Preferences
Preferences
Panel Grid Size:
18
Fine Move Delta:
1
Selection Handle Width:
6
Multiple Selection Key
Control
Shift
Colors
Root Window:
AliceBlue
AntiqueWhite
AntiqueWhite3
Selection Highlight:
red
red3
RosyBrown
Rehearsal Options
Rehearse Data Driven Objects
Type of Rehearsal
Min-Max
Max-Min
Max-Min-Max
Min-Max-Min
Cycles
Once
Repeated
Update Interval (milliseconds)
5000
All changes take effect immediately
Close
Help

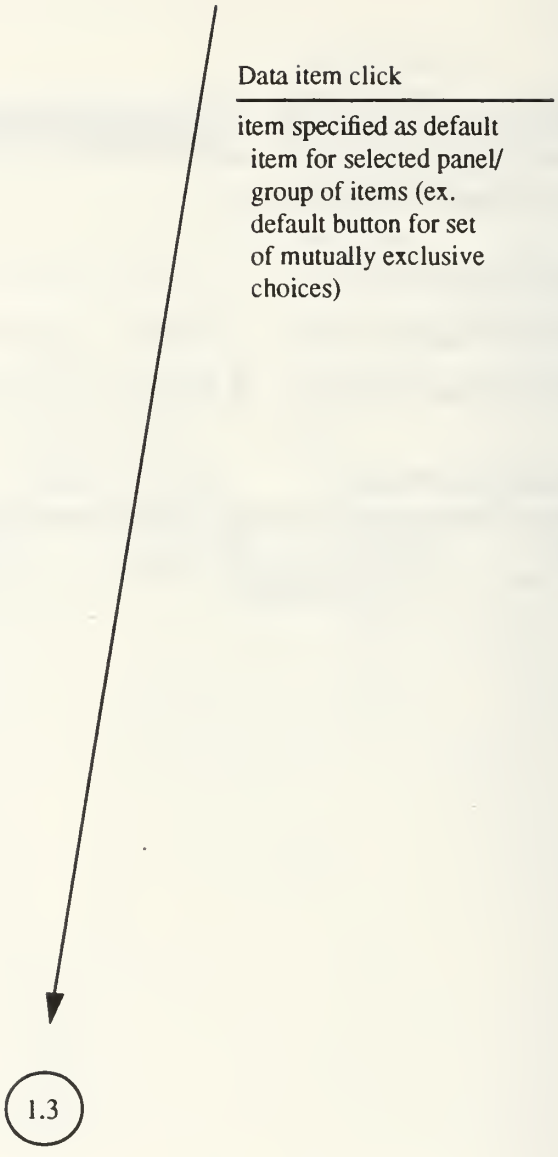

TAE Plus WorkBench V5.1
Resource File: test2.res
WorkBench Mode:
Move/Resize/Edit
Define Connections
Set Panel Default
Set Item Default
Undo Ctrl+u
New Panel Ctrl+p
New Item Ctrl+i
Current Selection :
File
Edit
Arrange
Auxiliary
Help
Connection Specification
Event (label1)
CURRENT Panel
Name : NoName01
Panel State After Connection
Preferred
Invisible
Iconic
No Change
Deleted
NEXT Panel
Name :
Panel State After Connection
Preferred
Invisible
Iconic
Visible
Deleted
Fast Iconic
TCL command:
OK
Cancel
Help


Item click
Connection Specification
window spawned
1.3.5.1
WAITING FOR
CONNECTION(S) DFN
"Cancel" click
window closed
(Panel state
after connect
executed, next
panel id, TCL cmd
defined/specified)
"OK" click
information
saved
source code
files updated
1.3


TAE Plus WorkBench V5.1
Resource File: test2.res
WorkBench Mode:
Move/Resize/Edit
Define Connections
Set Panel Default
Set Item Default
Undo Ctrl+u
New Panel Ctrl+p
New Item Ctrl+i
Current Selection :
File
Edit
Arrange
Auxiliary
Help


Push-button or icon item click
item specified as default item for selected panel
Non push-button or icon item click
user instructed that only push-buttons or icons can be declared as default panel items
1.3.6.1
WAITING FOR MESSAGE ACK
"NOTED" click
window closed
1.3


TAE Plus WorkBench V5.1
Resource File: test2.res
WorkBench Mode:
Move/Resize/Edit
Define Connections
Set Panel Default
Set Item Default
Undo Ctrl+u
New Panel Ctrl+p
New Item Ctrl+i
Current Selection :
File
Edit
Arrange
Auxiliary
Help


Data item click
item specified as default item for selected panel/ group of items (ex. default button for set of mutually exclusive choices)
1.3

APPENDIX C

TAE+ STATE-QUERIES-TABLE

This table provides a listing, catagorized by class of question, of the questions that arise at each state. Sellen and Nicol in [SELLE 90] cite an addtional type of question class, Descriptive. These types of questions are adequately addressed by the existing TAE+ application and therefore are not address here.

State Name/#	Goal-oriented *What things can I do?*	Procedural *How do I?*	Interpretive *Why did it happen?*	Navigative *Where am I?*
0 Waiting for TAE+ icon choice	Intro para on TAE+ - genl capabilities, use embedded help component to get descrip of Homeroom	CREATE AN INTERFACE HOW TO USE TAE+ HOW TO START TAE+		
1 Waiting for Resource File-name	Specify startup to be norm/icon/ fast icon Name file to open	HOW TO SET/CHG THE STARTUP MODE (?) HOW TO OPEN EXISTING/NEW FILE		
1.1 Waiting for Message Ack			TAE+ couldn't find named file, initiating new file	
1.2 Waiting for Homeroom Activation				
1.3 Waiting for WorkBench Mode/Operation Selection	Introductiory para re WB functionality, attention to embedded help for descriptive assistance	HOW TO ? INTERFACE PANEL ITEM HOW TO SET ITEM CONNECTIONS HOW TO SET STARTING/INITIAL PANEL HOW TO REHEARSE(TEST) INTERFACE HOW TO GENERATE SOURCE CODE HOW TO EXIT FILE/SYSTEM SAVE RESOURCE FILE EXIT W/O SAVE COPY PART OF FILE ? = create, delete, modify, save, find info re, move, chg defaults of		

State Name/#	Goal-oriented *What things can I do?*	Procedural *How do I?*	Interpretive *Why did it happen?*	Navigative *Where am I?*
1.3.1.1 Waiting for New Resource Filename	See state 1			
1.3.1.1.1 Waiting for Confirmation			User provided the filename of an existing file (as opposed to a new filename)	
1.3.1.2 Waiting for Existing Resource Filename	See state 1			
1.3.1.2.1 Waiting for Confirmation			See state 1.1	
1.3.1.3 Waiting for INCLUDE File name	Para describing INCLUDE functionality	HOW TO MERGE CONTENTS OF TWO FILES		Keeps file that INCLUDE was initiated from open
1.3.1.3.1 Waiting for Message Ack			TAE+ provided w/ invalid filename	

State Name/#	Goal-oriented	Procedural	Interpretive	Navigative
	What things can I do?	*How do I?*	*Why did it happen?*	*Where am I?*
1.3.1.4 Waiting for Resource File-name to Save to		HOW TO COPY PART/ALL OF INTERFACE TO DIFFERENT FILE		
1.3.2.1 Waiting for Panel Specs	Para describing concept of panel and things that can vary	HOW TO CREATE NEW/FIRST PANEL HOW TO SET THE PANEL CHARACTERISTICS		
1.3.2.1.1 Waiting for Panel Details		HOW TO CHANGE PANEL BORDER/ DIMENSIONS/TITLE BAR HOW TO LINK HELP TEXT/ICONS		
13.2.1.1.1 Waiting for Panel Dimen-sions				
1.3.2.2 Waiting for Item Specs	Para describing concept of item and things that can vary	HOW TO CREATE NEW ITEM HOW TO MODIFY/DELETE ITEM HOW TO DESIGNATE AN EVENT BUTTON		
1.3.2.2.1 Waiting for Valid Input Specs		HOW TO SPECIFY OPERATOR INPUT RESTRICTIONS		
1.3.2.2.2 Waiting for Item Dimensions		HOW TO SPECIFY SIZE ITEM		

State Name/#	Goal-oriented	Procedural	Interpretive	Navigative
	What things can I do?	*How do I?*	*Why did it happen?*	*Where am I?*
1.3.2.2.3 Waiting for Item Presentation Class	Para describing differences between item presentation classes	HOW TO SPECIFY AN ITEM (Type, custom characteristics, link to icon, size, spacing, etc.)		
1.3.2.3 Waiting for Group Modifications	Para explaining selection/group selection, add/delete/mods to groups of items and panels	HOW TO CHANGE ALL OR SELECT PANELS AND ITEMS HOW TO COPY/INSERT COPIES OF PANELS/ITEMS HOW TO SELECT GROUPS OF PANELS/ITEMS HOW TO DELETE SELECTED/ALL PANELS/ITEMS		
1.3.3.1 Waiting for Alignment Params	Para describing functionality of alignment tools, explanation of grid - toggle, snap-to	HOW TO SPECIFY ALIGNMENT SCHEME HOW TO APPLY/INVOKE ALIGNMENT ON ITEMS HOW TO ICONIFY SELECTED/ALL PANELS		
1.3.4.1 Waiting for Id of Startup Panel	Para describing concept of a "startup" panel	HOW TO SPECIFY INITIAL PANEL(S) HOW TO ADD TO/REMOVE FM PANEL LIST	"Initial panel(s)" list may not be complete/accurate/current	
1.3.4.2 Waiting for Generation Details	Para describing generating source code. Include info regarding stubs, available languages	HOW TO GENERATE SOURCE CODE	Resource file must be saved prior to attempting to generate source code	

State Name/#	Goal-oriented	Procedural	Interpretive	Navigative
	What things can I do?	*How do I?*	*Why did it happen?*	*Where am I?*
1.3.4.3 Waiting for User Interaction	Para describing concept of "Rehearse"	HOW TO TEST THE INTERFACE/ VIEW CONNECTIONS		
1.3.4.3.1 Waiting for WB Icon Selection			Need to click WB icon to continue	
1.3.4.4 Waiting for Terminal Command			Spawned a local terminal window from w/i TAE WB	
1.3.4.5 Waiting for WB Preferences	Para describing how to tailor/ customize WB characteristics	HOW TO DEFINE WB PARAMS HOW TO CHANGE WORKBENCH CHARACTERISTICS		
1.3.5.1 Waiting for Connection(s)	Para describing inter-panel connections functionality, panel states, use of TAE+ command language	HOW TO DEFINE CONNECTIONS BETW PANELS HOW TO CONFIG PANEL VISIBILITY HOW TO USE TAE+ COMMAND LANGUAGE		
1.3.6.1 Waiting for Message Ack	Para describing default panel items and default selections	HOW TO SPECIFY DEFAULT SELECTIONS HOW TO DEFINE DEFAULT PANELS/ITEMS	User can only designate push buttons or icons as default panel items (Set Panel Default)	

APPENDIX D

TAE+ CONTEXT-CLUES-TABLE

This table provides a listing, of the observable elements of each state and the questions that may be asked of the user to better determine the desired topic.

State Name/#	Observable	Questions to ask
0 Waiting for TAE+ icon choice	User's homeroom icon window (broad indicator only)	Is assistance needed on TAE in general or a specific component of TAE?
1 Waiting for Resource File-name	TAE+ WB: Resource filename Sub-W Designation of Startup mode: norm, icon, fast icon	Resource file name or Startup mode request?
1.1 Waiting for Message Ack	Message Sub-W	Resource file name request?
1.2 Waiting for Homeroom Activation	Previous window closes, homeroom iconified (this happens when cancel entered at Resource file name entry window)	TAE general/getting started request? Resource file name request?
1.3 Waiting for WorkBench Mode/Operation Selection	WorkBench menu window(TAE Plus WorkBench V5.1 Sub-W) (broad indicator only)	Is assistance needed on the WorkBench in general, a specific mode, or a specific command menu?

State Name/#	Observable	Questions to ask
1.3.1.1 Waiting for New Resource Filename	New File Sub-W If already have an application open -> may be trying to merge two existing interfaces, start fresh with or without saving current If currently have a new application open -> may be trying to find previously created interface	File save, with or without saving current request? Merging two interfaces? Starting new interface/restarting existing?
1.3.1.1.1 Waiting for Confirmation	(User provided existing vs. new filename) New Sub-W	"
1.3.1.2 Waiting for Existing Resource Filename	Open File Sub-W If already have an application open -> may be trying to merge two existing interfaces, start fresh with or without saving current If currently have a new application open -> may be trying to find previously created interface	"
1.3.1.2.1 Waiting for Confirmation	(User provided new vs. existing filename) Open Sub-W	"
1.3.1.3 Waiting for INCLUDE File name	Include File Sub-W	Merging two interfaces/Adding to existing interface?
1.3.1.3.1 Waiting for Message Ack	Message Sub-W	"
1.3.1.4 Waiting for Resource Filename to Save to	Save As Sub-W	File save to new file/update existing file?

State Name/#	Observable	Questions to ask
1.3.2.1 Waiting for Panel Specs	Panel Specification Sub-W	Is assistance needed about Panels in general or specific component of the panel specifications?
1.3.2.1.1 Waiting for Panel Details	Panel Details Sub-W	General panel details information or specifics to panel frame, panel titlebar, &/or panel state? Linking to/editing existing built-in help Linking to/editing existing icon?
13.2.1.1.1 Waiting for Panel Dimensions	Dimension Specification Sub-W	Request for information re placement of the panel on the screen or on the default/changing the default size of the panel?
1.3.2.2 Waiting for Item Specs	Item Specification Sub-W	Is assistance needed re Items in general or specific component of the item specifications?
1.3.2.2.1 Waiting for Valid Input Specs	Candidate Strings Sub-W/Integer Constraints/ Real Constraints Sub-W (depending upon the item's declared data type) (from Set Constraints button)	Input restrictions supported in general or specifics of one type?
1.3.2.2.2 Waiting for Item Dimensions	Dimension Specification Sub-W(from Specify Dimensions button)	Request information on placement of item on panel or default/changing the default of item size?
1.3.2.2.3 Waiting for Item Presentation Characteristics	Presentation Sub-W Specific to the category/type of item (depending upon the item's declared category/type) (from Specify Details button)	Request information on details of cat/type in general or specifics?

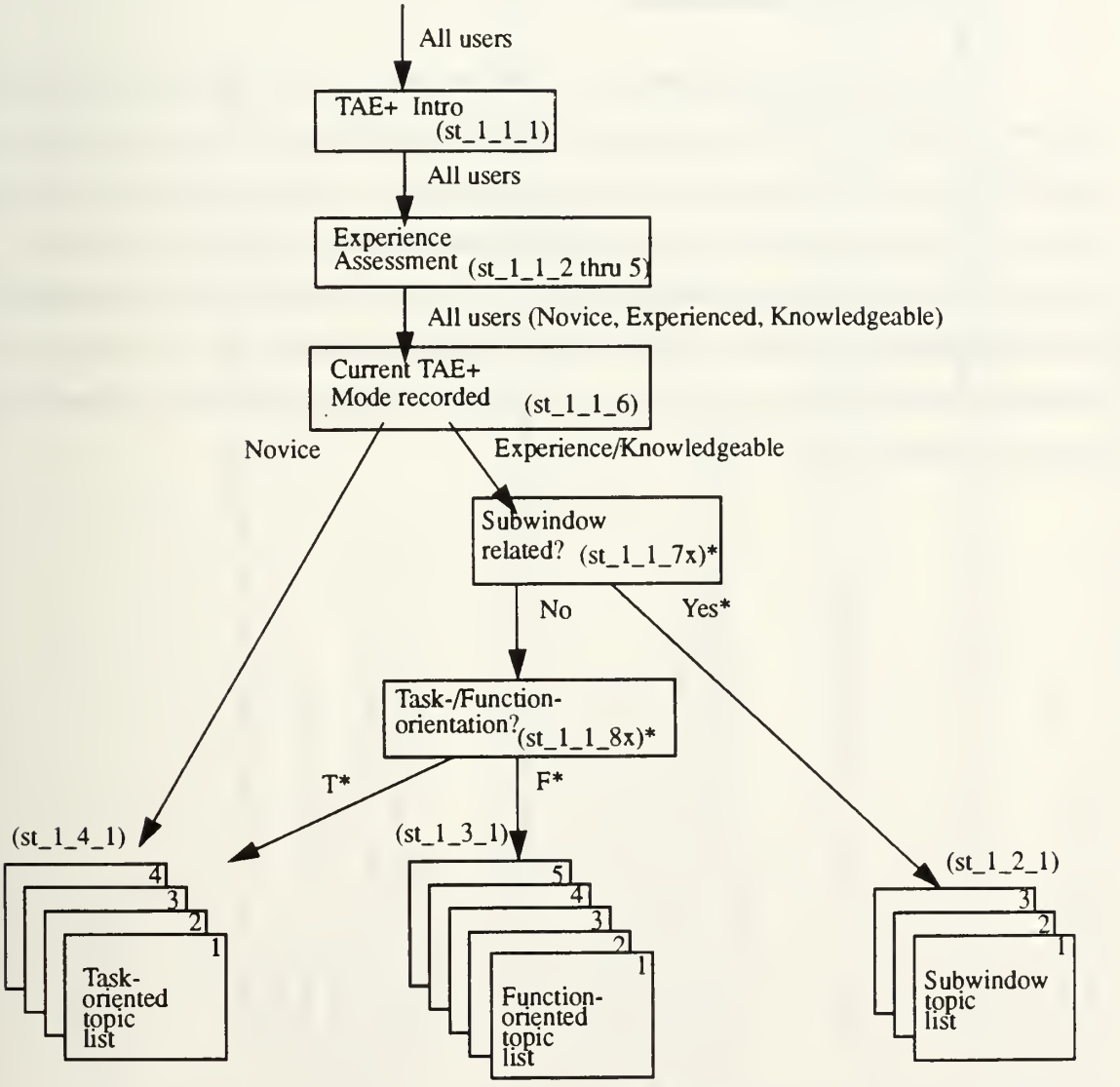
State Name/#	Observable	Questions to ask
1.3.2.3 Waiting for Group Modifications	Group Modification Sub-W may have selected more than one item/panel in error may be trying to delete interface	Request for modifying all/some/one item(s); all/some/one panel(s)? Deleting all?
1.3.3.1 Waiting for Alignment Params	Current Selection Alignment Sub-W	Request for general alignment information?
Use of Arrange, Align	None	Request for how to set desired alignment? How to change defaults?
Use of Arrange, Toggle Grid	None	Request for general information re use of Grid, turning on/off? Changing default spacing betw grids, Snap to grid feature?
Use of Arrange, Snap to Grid	None	Request for general info re use of Snap, changing defaults of grid size or allowable delta betw item and grid line?
Use of Arrange, Iconify	None	Request for information re iconifying panels, placement of panel/icon on screen?
1.3.4.1 Waiting for Id of Startup Panel	Specify Initial Panels Sub-W	Request for info re how to identify panel as initial panel, how to identify multiple panels as startup panels, adding panels to/removing panels from existing list? General information re Startup panels
1.3.4.2 Waiting for Generation Details	Application Generation Sub-W (from Code Generation selection)	Request for information re source language, file structure, diagnostic messages, event handlers

State Name/#	Observable	Questions to ask
1.3.4.3 Waiting for User Interaction Use of Aux, Rehearsal	(from Rehearse selection) wbrehearse Sub-W	Request info about rehearse results?
1.3.4.3.1 Waiting for WB Icon Selection	(from close of rehearse window) WB icon Sub-W	Request for info re resumption/termination of WorkBench or TAE+?
1.3.4.4 Waiting for Terminal Command	(from Create Terminal selection) Additional xterm Sub-W	? supportable?
1.3.4.5 Waiting for WB Preferences	WorkBench Preferences Sub-W	Request for info re general parameters that can be set? Info specific to panel/item(s), rehearsal?
1.3.5.1 Waiting for Connection(s)	(from Define Connections WB Mode selection) Connection Specification Sub-W	Request for general info re connections and their usage? For usage information of TCL Command entry? Std entries? For panel state information?
1.3.6.1 Waiting for Message Ack	(from Set Panel Default WB Mode selection when no panel identified) Message Sub-W	Request for general info re identifying default panel items, usage of same?

APPENDIX E

TAE+ HELP SYSTEM PROGRAM LISTING

This appendix provides the program script for TAE+ Help. Figure 1 below, provides a pictorial representation of the system setup process, establishing the initial user experience level and menu structure choice.



*Menu access point dependent upon current TAE+ mode

The general format of the program listing is to declare the state being addressed, specify the actions that occur during each instance of the state and detail all the possible transitions out of the state and the resulting actions of each transition. For example:

```
(st_1_1_1,
 ((0, clear),
  (0, write, " Welcome to the TAE+ Help System.. Have you used TAE+ in the past? (Please
enter Y or N)",
  (9, input, keyboard)),
((15 seconds & (past st_1_1_1 wait), ),
  (15 seconds, (assert, wait), st_1_1_1),
  ("YES"|"Yes"|"yes"|"Y"|"y", (assert, TAE_user), st_1_1_4),
  ("NO"|"No"|"no"|"N"|"n", (assert, non_TAE_user), st_1_1_2),
 st_1_1_1))     /*unexpected ans*/
```

The above code defines state st_1_1_1. The first action (0, clear) clears the screen, the second (0, write, " Welcome to the TAE+ Help System...") writes a message to the screen and the final action (9, input, keyboard) enables the keyboard to receive input from the user. The next two lines provide 30 seconds, in two 15 second increments, for the user to respond to the question. If the user does not answer within the alloted 30 seconds, control transfers to state st_1_1_20 which informs the user that an assumption has been made regarding his experience with using TAE+. If the user provides a "yes" or "no" answer, control transfers to subsequent state st_1_1_4 or st_1_1_2 respectively; any other answer results in control looping back into the state st_1_1_1.

```
(menu "TAE+ Help System"
 "Introduction" -> st_1_1_1)

(st_1_1_1,
 ((0, clear),
  (0, write, "  Welcome to the TAE+ Help System.  Have you used TAE+ in the past?  (Please enter Y or N)"),
  (9, input, keyboard)),
 ((15 seconds & (past st_1_1_1 wait), st_1_1_20),
  (15 seconds, (assert, wait), st_1_1_1),
  ("YES"|"Yes"|"yes"|"Y"|"y", (assert, TAE_user), st_1_1_4),
  ("NO"|"No"|"no"|"N"|"n", (assert, non_TAE_user), st_1_1_2),
  st_1_1_1))  /*unexpected ans*/

(st_1_1_2,
 ((0, write, "  Some of the fundamental principles of TAE+ are in some ways similar to Hypertext.  Have you used Hypertext in the past?  (Please
enter Y or N)"),
  (9, input, keyboard)),
 ((15 seconds & (past st_1_1_2 wait), st_1_1_30),
  (15 seconds, (assert, wait), st_1_1_2),
  ("YES"|"Yes"|"yes"|"Y"|"y", (assert, hyper_user), st_1_1_3),
  ("NO"|"No"|"no"|"N"|"n", st_1_1_100), /*set experienced level to Novice*/
  st_1_1_2))  /*unexpected ans*/

(st_1_1_3,
 ((0, write, "  ...help info in terms of TAE+ panel equivalence to hyper background and TAE+ item equal to hyper buttons, etc..."), (0, pause, 5)),
 st_1_1_100)

(st_1_1_4,
 ((0, write, "  Have you used either the TAE+ Command Language (TCL) to update objects or detail TAE+ generated source program event stubs?
(Please enter Y or N)"),
  (9, input, keyboard)),
 ((15 seconds & (past st_1_1_4 wait), st_1_1_25),
  (15 seconds, (assert, wait), st_1_1_4),
  ("YES"|"Yes"|"yes"|"Y"|"y", st_1_1_300),  /*set experienced level to Knowledgeable*/
```

```
  ("NO"|"No"|"no"|"N"|"n", st_1_1_5),
  st_1_1_4))  /*unexpected ans*/

(st_1_1_5,
  ((0, write, "  Have you used either Define Connections to link panels or Set Panel Default/Item Panel Default functions?  (Please enter Y or N)"),
  (9, input, keyboard)),
  ((15 seconds & (past st_1_1_5 wait), st_1_1_30),
  (15 seconds, (assert, wait), st_1_1_5),
  ("YES"|"Yes"|"yes"|"Y"|"y", st_1_1_200), /*set experienced level to Experienced*/
  ("NO"|"No"|"no"|"N"|"n", st_1_1_100),   /*set experienced level to Novice*/
  st_1_1_5))   /*unexpected ans*/

/* Collects initial application mode from Experienced/Knowledgeable user */
/* See st_1_1_8 for Novice user */
(st_1_1_6, ((0, retract, st_1_1_6, edit_mode), (0, retract, st_1_1_6, connect_mode),
  (0, retract, st_1_1_6, panel_mode), (0, retract, st_1_1_6, item_mode),
  (0, retract, st_1_1_7, edit_mode), (0, retract, st_1_1_7, connect_mode),
  (0, retract, st_1_1_7, panel_mode), (0, retract, st_1_1_7, item_mode),
  (0, retract, st_1_1_8, edit_mode), (0, retract, st_1_1_8, connect_mode),
  (0, retract, st_1_1_8, panel_mode), (0, retract, st_1_1_8, item_mode),
  (0, write, " Select the number representing the current WorkBench Mode:
  1. Move/Resize/Edit
  2. Define Connections
  3. Set Panel Default
  4. Set Item Default
  (Please enter 1, 2, 3 or 4)"),
  (9, input, keyboard)),
  ((15 seconds & (past st_1_1_6 wait), st_1_1_45),
  (15 seconds, (assert, wait), st_1_1_6),
  ("1"|"One"|"one"|"ONE", (assert, edit_mode), st_1_1_6_1_1),
  ("2"|"Two"|"two"|"TWO", (assert, connect_mode), st_1_1_6_2_1),
  ("3"|"Three"|"three"|"THREE", (assert, panel_mode), st_1_1_6_3_1),
  ("4"|"Four"|"four"|"FOUR", (assert, item_mode), st_1_1_6_4_1),
  st_1_1_6))  /*unexpected ans*/

(st_1_1_6_1_1,  /* for Move/Resize/Edit mode */
```

```
  ((0, write, "  Is your help request related to a particular TAE+ sub-window?  (Please enter Y or N) "),
   (9, input, keyboard)),
   ((15 seconds & (past st_1_1_6_1_1 wait), st_1_1_35),
   (15 seconds, (assert, wait), st_1_1_6_1_1),
   ("YES"|"Yes"|"yes"|"Y"|"y", st_1_2_1),
   ("NO"|"No"|"no"|"N"|"n", st_1_1_6_1_2),
   st_1_1_6_1_1))    /*unexpected ans*/

 (st_1_1_6_1_2,
   ((0, write, "  Order the help topics by TAE+ Task(T) or alphabetically by Function(F)?  (Please enter T or F) "),
   (9, input, keyboard)),
   ((15 seconds & (past st_1_1_6_1_2 wait), st_1_1_40),
   (15 seconds, (assert, wait), st_1_1_6_1_2),
   ("T"|"t", st_1_4_1),
   ("F"|"f", st_1_3_1),
   st_1_1_6_1_2))  /*unexpected ans*/

 (st_1_1_6_2_1,  /* for Define Connections mode */
   ((0, write, "  Is your help request related to a particular TAE+ sub-window?  (Please enter Y or N) "),
   (9, input, keyboard)),
   ((15 seconds & (past st_1_1_6_2_1 wait), st_1_1_35),
   (15 seconds, (assert, wait), st_1_1_6_2_1),
   ("YES"|"Yes"|"yes"|"Y"|"y", st_1_2_150),
   ("NO"|"No"|"no"|"N"|"n", st_1_1_6_2_2),
   st_1_1_6_2_1))    /*unexpected ans*/

 (st_1_1_6_2_2,
   ((0, write, "  Order the help topics by TAE+ Task(T) or alphabetically by Function(F)?  (Please enter T or F) "),
   (9, input, keyboard)),
   ((15 seconds & (past st_1_1_6_2_2 wait), st_1_1_40),
   (15 seconds, (assert, wait), st_1_1_6_2_2),
   ("T"|"t", st_1_4_400),
   ("F"|"f", st_1_3_1),
   st_1_1_6_2_2))  /*unexpected ans*/

 (st_1_1_6_3_1,  /* for Set Panel Default mode */
```

```
((0, write, "  Is your help request related to a particular TAE+ sub-window?  (Please enter Y or N) "),
 (9, input, keyboard)),
((15 seconds & (past st_1_1_6_3_1 wait), st_1_1_35),
 (15 seconds, (assert, wait), st_1_1_6_3_1),
 ("YES"|"Yes"|"yes"|"Y"|"y", st_1_2_150),
 ("NO"|"No"|"no"|"N"|"n", st_1_1_6_3_2),
 st_1_1_6_3_1))    /*unexpected ans*/

(st_1_1_6_3_2,
 ((0, write, "  Order the help topics by TAE+ Task(T) or alphabetically by Function(F)?  (Please enter T or F) "),
  (9, input, keyboard)),
 ((15 seconds & (past st_1_1_6_3_2 wait), st_1_1_40),
  (15 seconds, (assert, wait), st_1_1_6_3_2),
  ("T"|"t", st_1_4_300),
  ("F"|"f", st_1_3_1),
  st_1_1_6_3_2))  /*unexpected ans*/

(st_1_1_6_4_1,  /* for Set Item Default mode */
 ((0, write, "  Is your help request related to a particular TAE+ sub-window?  (Please enter Y or N) "),
  (9, input, keyboard)),
 ((15 seconds & (past st_1_1_6_4_1 wait), st_1_1_35),
  (15 seconds, (assert, wait), st_1_1_6_4_1),
  ("YES"|"Yes"|"yes"|"Y"|"y", st_1_2_150),
  ("NO"|"No"|"no"|"N"|"n", st_1_1_6_4_2),
  st_1_1_6_4_1))    /*unexpected ans*/

(st_1_1_6_4_2,
 ((0, write, "  Order the help topics by TAE+ Task(T) or alphabetically by Function(F)?  (Please enter T or F) "),
  (9, input, keyboard)),
 ((15 seconds & (past st_1_1_6_4_2 wait), st_1_1_40),
  (15 seconds, (assert, wait), st_1_1_6_4_2),
  ("T"|"t", st_1_4_300),
  ("F"|"f", st_1_3_300),
  st_1_1_6_4_2))  /*unexpected ans*/
```

```
/* Collects application mode in use when mode change is detected during use session */
/* See st_1_1_6 and st_1_1_8 for initial setting mechanisms*/
(st_1_1_7, ((0, retract, st_1_1_6, edit_mode), (0, retract, st_1_1_6, connect_mode),
  (0, retract, st_1_1_6, panel_mode), (0, retract, st_1_1_6, item_mode),
  (0, retract, st_1_1_7, edit_mode), (0, retract, st_1_1_7, connect_mode),
  (0, retract, st_1_1_7, panel_mode), (0, retract, st_1_1_7, item_mode),
  (0, retract, st_1_1_8, edit_mode), (0, retract, st_1_1_8, connect_mode),
  (0, retract, st_1_1_8, panel_mode), (0, retract, st_1_1_8, item_mode),
  (0, write, " Select the appropriate number to reset the current WorkBench Mode:
  1. Move/Resize/Edit
  2. Define Connections
  3. Set Panel Default
  4. Set Item Default
  (Please enter 1, 2, 3 or 4)"),
  (9, input, keyboard)),
  ((15 seconds & (past st_1_1_7 wait), st_1_1_45),
  (15 seconds, (assert, wait), st_1_1_7),
  ("1"|"One"|"one"|"ONE", (assert, edit_mode), st_1_1_7_1),
  ("2"|"Two"|"two"|"TWO", (assert, connect_mode), st_1_1_7_2),
  ("3"|"Three"|"three"|"THREE", (assert, panel_mode), st_1_1_7_3),
  ("4"|"Four"|"four"|"FOUR", (assert, item_mode), st_1_1_7_4),
  st_1_1_7))  /*unexpected ans*/

(st_1_1_7_2,
  ((0, write, "...WorkBench mode has been reset to 'Define Connections'..."),
  (0, pause, 2)),
  ((past st_1_3_45 active), st_1_3_45_1))

/* Collects initial application mode from Novice user */
/* See st_1_1_6 for Experienced/Knowledgeable user */
(st_1_1_8, ((0, write, " Select the number representing the current WorkBench Mode:
  1. Move/Resize/Edit
  2. Define Connections
  3. Set Panel Default
  4. Set Item Default
  5. I don't know or I have not logged into TAE+ yet.
```

```
 (Please enter 1, 2, 3, 4 or 5)"),
  (9, input, keyboard)),
 ((15 seconds & (past st_1_1_8 wait), st_1_1_50),
  (15 seconds, (assert, wait), st_1_1_8),
  ("1"|"One"|"one"|"ONE", (assert, edit_mode), st_1_4_1),
  ("2"|"Two"|"two"|"TWO", (assert, connect_mode), st_1_4_1),
  ("3"|"Three"|"three"|"THREE", (assert, panel_mode), st_1_4_1),
  ("4"|"Four"|"four"|"FOUR", (assert, item_mode), st_1_4_1),
  ("5"|"Five"|"five"|"FIVE", (assert, edit_mode), st_1_4_1),  /*default to task menu to assist novice user*/
  st_1_4_1))  /*unexpected ans*/

(st_1_1_20,
 ((0, write, "  You have not replied or have entered an invalid answer.  It is assumed that you are not a TAE user."), (0, pause, 5)),
 st_1_1_2)

(st_1_1_25,
 ((0, write, "  You have not replied or have entered an invalid answer.  It is assumed that you have not used TAE extensively."), (0, pause, 5)),
 st_1_1_5)

(st_1_1_30,
 ((0, write, "  You have not replied or have entered an invalid answer.  It is assumed that you are not an experienced user.  User experience level is
set to 'Novice'"), (0, pause, 5)),
 st_1_1_100)

(st_1_1_35,
 ((0, write, "  You have not replied or have entered an invalid answer.  It is assumed this is not a sub-window related request."), (0, pause, 5)),
 st_1_1_8)

(st_1_1_40,
 ((0, write, "  You have not replied or have entered an invalid answer.  It is assumed you wish the topics to be ordered by Task."), (0, pause, 5)),
 st_1_4_1)   /*Presented w/ Task-oriented topic list*/

(st_1_1_45,
 ((0, write, "  You have not replied or have entered an invalid answer.  It is assumed you are in Move/Resize/Edit mode."), (0, pause, 5)),
 st_1_2_1)   /*User is experienced, thus presented w/ function-oriented topic list*/
```

```
(st_1_1_50,
 ((0, write, "  You have not replied or have entered a default answer."), (0, pause, 5)),
 st_1_4_1)   /*User is not experienced, thus presented w/ task-oriented topic list*/

(st_1_1_100,
 ((0, clear),
  (0, retract, st_1_1_100, novice_user), (0, retract, st_1_1_200, exper_user), (0, retract, st_1_1_300, knowl_user),
  (0, write, "...experience level set to NOVICE..."),
  (6, write, "please click the mouse ...")),
 /* No active states during the initial setup - falls thru to here */
  ((assert, novice_user), st_1_1_8))

(st_1_1_200,
 ((0, clear),
  (0, retract, st_1_1_100, novice_user), (0, retract, st_1_1_200, exper_user), (0, retract, st_1_1_300, knowl_user),
  (0, write, "...experience level set to EXPERIENCED..."),
  (6, write, "please click the mouse ...")),
  /* from 1_4_3 to 1_4_11 downgrade exper level from knowl_user to exper_user*/
 (((past st_1_4_3 activeone), (assert, exper_user), st_1_4_3),
  ((past st_1_4_3 activetwo), (assert, exper_user), st_1_4_3),
  ((past st_1_4_4 active), (assert, exper_user), st_1_4_4),
  ((past st_1_4_5 active), (assert, exper_user), st_1_4_5),
  ((past st_1_4_60 active), (assert, exper_user), st_1_4_60),
  ((past st_1_4_6 activeone), (assert, exper_user), st_1_4_6),
  ((past st_1_4_6 activetwo), (assert, exper_user), st_1_4_6),
  ((past st_1_4_7 active), (assert, exper_user), st_1_4_7),
  ((past st_1_4_8 active), (assert, exper_user), st_1_4_8),
  ((past st_1_4_9 active), (assert, exper_user), st_1_4_9),
  ((past st_1_4_10 active), (assert, exper_user), st_1_4_10),
  ((past st_1_4_11 active), (assert, exper_user), st_1_4_11),
  /* from 1_4_38 to 1_4_46 upgrade exper level from novice_user to exper_user.  Some left out for brevity.*/
  ((past st_1_4_38 active), (assert, exper_user), st_1_4_38),
  ((past st_1_4_39 active), (assert, exper_user), st_1_4_39),
 /* No active states during the initial setup - falls thru to here */
  (assert, exper_user), st_1_1_6))
```

```
(st_1_1_300,
 ((0, clear),
  (0, retract, st_1_1_100, novice_user), (0, retract, st_1_1_200, exper_user), (0, retract, st_1_1_300, knowl_user),
  (0, write, "...experience level set to KNOWLEDGEABLE..."),
  (6, write, "please click the mouse ...")),
  /* from 1_4_47 to 1_4_49 upgrade exper level from exper_user to knowl_user*/
 (((past st_1_4_47 active), (assert, knowl_user), st_1_4_47),
  ((past st_1_4_48 active), (assert, knowl_user), st_1_4_48),
  ((past st_1_4_49 active), (assert, knowl_user), st_1_4_49),
  /* No active states during the initial setup - falls thru to here */
  (assert, knowl_user), st_1_1_6))

(st_1_2_1,
 ((0, clear),
  (0, write, "   1. Sub-Window List
              2. TAE+ WorkBench:  Resource Filename
              3. FILE
              4.  New File
              5.  Open File
              6.  Include File
              7.  Save As File
              8. PANEL/ITEM/OTHER COMMANDS

  (Enter topic number)"),
  (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
  (9, input, mouse&key)),
 ((click-left, st_1_2_1),    /*left button calls for sub-w list*/
  (click-middle, st_1_3_450),  /*middle button calls for function list*/
  (click-right, st_1_4_1),   /*right button calls for task list*/
  ("1"|"One"|"one"|"ONE", st_1_2_100),
  ("2"|"Two"|"two"|"TWO", st_1_2_2),
  ("3"|"Three"|"three"|"THREE", st_1_2_3),
  ("4"|"Four"|"four"|"FOUR", st_1_2_4),
  ("5"|"Five"|"five"|"FIVE", st_1_2_5),
  ("8"|"N"|"n"|"Eight"|"eight"|"EIGHT", st_1_2_150),
```

```
  ("6"|"7", st_1_2_225),
  st_1_2_225))

(st_1_2_2,
  (0, clear),
  (assert, shown), st_1_4_3)

(st_1_2_3,
  (0, clear),
  (assert, shown), st_1_3_60)

(st_1_2_4,
  (0, clear),
  (assert, shown), st_1_4_5)

(st_1_2_5,
  (0, clear),
  (assert, shown), st_1_4_6)

(st_1_2_8,
  (0, clear),
  (assert, shown), st_1_3_43)

(st_1_2_9,
  (0, clear),
  st_1_2_40)

(st_1_2_10,
  (0, clear),
  st_1_2_50)

(st_1_2_11,
  (0, clear),
  st_1_2_60)

(st_1_2_12,
```

```
 (0, clear),
 (assert, shown), st_1_3_23)

(st_1_2_13,
 (0, clear),
 st_1_2_70)

(st_1_2_29,
 (0, clear),
 (assert, shown), st_1_3_45)

(st_1_2_40,
 ((0, clear),
  (0, write, "  Select topic number:
    1. Panel name
    2. Panel title
    3. Panel details (...subsequent menu...)
    4. Other (...provides task-oriented menu...)

  (Enter 1, 2, 3 or 4) "),
  (8, write, "Left-Middle-Right mouse buttons provide subwindow, function, and task oriented topic lists respectively."),
  (9, input, mouse&key)),
 ((click-left, st_1_2_150),    /*left button calls for sub-w list*/
  (click-middle, st_1_3_400),  /*middle button calls for function list*/
  (click-right, st_1_4_1),   /*right button calls for task list*/
  ("1"|"One"|"one"|"ONE", (assert, shown), st_1_4_13),
  ("2"|"Two"|"two"|"TWO", (assert, shown), st_1_4_14),
  ("3"|"Three"|"three"|"THREE", st_1_2_50),
  ("4"|"Four"|"four"|"FOUR", st_1_4_1),
  st_1_2_250))

(st_1_2_50,
 ((0, clear),
  (0, write, "  Select topic number:
    1. Frame
    2. Titlebar
```

```
  3. Panel State
  4. Panel Dimensions (...subsequent menu...)
  5. Panel Help file
  6. Panel icon file
  7. Other (...provides task-oriented menu...)

 (Enter 1, 2, 3, 4, 5, 6 or 7)"),
 (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
 (9, input, mouse&key)),
((click-left, st_1_2_150),   /*left button calls for sub-w list*/
 (click-middle, st_1_3_400), /*middle button calls for function list*/
 (click-right, st_1_4_300),  /*right button calls for task list*/
 ("1"|"One"|"one"|"ONE", (assert, shown), st_1_2_275),      /* will go to st_1_4_16 eventually */
 ("2"|"Two"|"two"|"TWO", (assert, shown), st_1_4_14),
 ("3"|"Three"|"three"|"THREE", (assert, shown), st_1_2_275), /* will go to st_1_4_17 eventually */
 ("4"|"Four"|"four"|"FOUR", st_1_2_60),
 ("5"|"Five"|"five"|"FIVE", (assert, shown), st_1_2_275),    /* will go to st_1_4_19 eventually */
 ("6"|"Six"|"six"|"SIX", (assert, shown), st_1_2_275),      /* will go to st_1_4_20 eventually */
 ("7"|"Seven"|"seven"|"SEVEN", st_1_4_1),  /*Assumes non-context related*/
 st_1_2_275))

(st_1_2_60,
 ((0, clear),
  (0, write, "  Select topic number:
    1. Panel Origin
    2. Panel Size
    3. Other (...provides task_oriented menu...)

   (Enter 1, 2 or 3)"),
  (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
  (9, input, mouse&key)),
 ((click-left, st_1_2_150),   /*left button calls for sub-w list*/
  (click-middle, st_1_3_400), /*middle button calls for function list*/
  (click-right, st_1_4_300),  /*right button calls for task list*/
  ("1"|"One"|"one"|"ONE", (assert, shown), st_1_4_21),
  ("2"|"Two"|"two"|"TWO", (assert, shown), st_1_4_22),
```

```
  ("3"|"Three"|"three"|"THREE", st_1_4_1))) /*Assumes non-context related*/

(st_1_2_70,
 ((0, clear),
  (0, write, "  Select topic number:
  1. Item name
  2. Item constraints
  3. Item dimensions (...subsequent menu...)
  4. Item details
  5. Other (...provides task-oriented menu...)

  (Enter 1, 2, 3, 4 or 5)"),
  (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
  (9, input, mouse&key)),
 ((click-left, st_1_2_150),    /*left button calls for sub-w list*/
  (click-middle, st_1_3_300),  /*middle button calls for function list*/
  (click-right, st_1_4_400),   /*right button calls for task list*/
  ("1"|"One"|"one"|"ONE", st_1_4_26),
  ("2"|"Two"|"two"|"TWO", st_1_2_325),       /* will go to st_1_4_37 eventually */
  ("3"|"Three"|"three"|"THREE", st_1_2_80),
  ("4"|"Four"|"four"|"FOUR", st_1_2_325),    /* will go to st_1_4_42 eventually */
  ("5"|"Five"|"five"|"FIVE", st_1_4_1),
  st_1_2_325))

(st_1_2_80,
 ((0, clear),
  (0, write, "  Select topic number:
  1. Item Origin
  2. Item Size
  3. Other (...provides task_oriented menu...)

  (Enter 1, 2 or 3)"),
  (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
  (9, input, mouse&key)),
 ((click-left, st_1_2_150),    /*left button calls for sub-w list*/
  (click-middle, st_1_3_350),  /*middle button calls for function list*/
```

```
  (click-right, st_1_4_400),  /*right button calls for task list*/
  ("1"|"One"|"one"|"ONE", (assert, shown), st_1_2_350),  /* will go to 1_4_38 eventually */
  ("2"|"Two"|"two"|"TWO", (assert, shown), st_1_2_350),  /* will go to 1_4_39 eventually */
  ("3"|"Three"|"three"|"THREE", st_1_4_1), /*Assumes non-context related*/
  st_1_2_350))

(st_1_2_100,
 ((0, write, "  TITLE:  Sub-Window List.
 This menu or topic list provides the user with a list of the TAE+ windows a user observes while using the application.  Therefore, this can be used
by the user to quickly access window-related topics."),
  (6, write, "Type 'C' when ready to continue.  You will automatically be returned to the subwindow list."),
  (9, input, keyboard)),
 ((click-continue, (assert, shown), st_1_2_1),
  ("C"|"c", (assert, shown), st_1_2_1)))

(st_1_2_150,
 ((0, clear),
  (0, write, "   8. PANEL                    12. ITEM
   9.  Panel Specification                     13.  Item Specification
   10.  Panel Details                          14.  Item Constraints
   11.  Dimension Specification                   15.    String type constraints
                                               16.    Integer type constraints
                                               17.    Real type constraints
                                               18.  Dimension Specification
                                               19.  Presentation Details

 (Enter topic number, 'N' for next subwindow page or 'P' for previous subwindow page)"),
  (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
  (9, input, mouse&key)),
 ((click-left, st_1_2_1),     /*left button calls for sub-w list*/
  (click-middle, st_1_3_300),  /*middle button calls for function list*/
  (click-right, st_1_4_1),  /*right button calls for task list*/
  ("n"|"N", st_1_2_200),
  ("P"|"p", st_1_2_1),
  ("14"|"15"|"16"|"17"|"18"|"19", st_1_2_375),
  ("8"|"Eight"|"eight"|"EIGHT", st_1_2_8),
```

```
 ("9"|"Nine"|"nine"|"NINE", st_1_2_9),
 ("10"|"Ten"|"ten"|"TEN", st_1_2_10),
 ("11"|"Eleven"|"eleven"|"ELEVEN", st_1_2_11),
 ("12"|"Twelve"|"twelve"|"TWELVE", st_1_2_12),
 ("13"|"Thirteen"|"thirteen"|"THIRTEEN", st_1_2_13),
 st_1_2_375))

(st_1_2_200,
 ((0, clear),
  (0, write, "  20. DUPLICATE                    24. AUXILIARY
                21.  Group Modification          25.  Specify Initial Panels
                22. ALIGNMENT                    26.  Application Generation
                23.  Current Selection Alignment 27.  Terminal
                                                 28.  WB Preferences
                                                 29.  Connections Specification
  (Enter topic number or 'P' for previous subwindow page)"),
  (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
  (9, input, mouse&key)),
 ((click-left, st_1_2_1),     /*left button calls for sub-w list*/
  (click-middle, st_1_3_1),   /*middle button calls for function list*/
  (click-right, st_1_4_400),  /*right button calls for task list*/
  ("P"|"p", st_1_2_150),
  ("29"|"Twentynine"|"Twentynine"|"TWENTYNINE", st_1_2_29),
  ("20"|"21"|"22"|"23"|"24"|"25"|"26"|"27"|"28", st_1_2_400),
  st_1_2_400))

(st_1_2_225,
 ((0, write, "  Selected option not yet implemented."),
  (6, write, "Returning you to your originating menu page....."),
  (0, pause, 5)),
 st_1_2_1)

(st_1_2_250,
 ((0, write, "  Selected option not yet implemented."),
  (6, write, "Returning you to your originating menu page..."),
  (0, pause, 5)),
```

```
 st_1_2_40)

(st_1_2_275,
 ((0, write, "  Selected option not yet implemented."),
  (6, write, "Returning you to your originating menu page..."),
  (0, pause, 5)),
 st_1_2_50)

(st_1_2_325,
 ((0, write, "  Selected option not yet implemented."),
  (6, write, "Returning you to your originating menu page..."),
  (0, pause, 5)),
 st_1_2_70)

(st_1_2_350,
 ((0, write, "  Selected option not yet implemented."),
  (6, write, "Returning you to your originating menu page..."),
  (0, pause, 5)),
 st_1_2_80)

(st_1_2_375,
 ((0, write, "  Selected option not yet implemented."),
  (6, write, "Returning you to your originating menu page..."),
  (0, pause, 5)),
 st_1_2_150)

(st_1_2_400,
 ((0, write, "  Selected option not yet implemented."),
  (6, write, "Returning you to your originating menu page..."),
  (0, pause, 5)),
 st_1_2_250)

(st_1_3_1,
 ((0,clear),
  (0, write, "   1. Function-oriented Topic List        9. Defaults
```

2. Aligning objects
3. Changing alignment delta
4. Specifying alignment params
5. Applying a command/action
6. Canceling a command/action
7. Changing the panel dim/loc
8. Changing the item dim/loc
10. Setting default panel
11. Setting item default value
12. Defining panel connections
13. Quit command
14. Setting panel state/visib
15. TCL command use

(Enter topic number or 'N' for next function-oriented page)"),
(8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
(9, input, mouse&key)),
((click-left, st_1_2_150), /*left button calls for sub-w list*/
(click-middle, st_1_3_1), /*middle button calls for function list*/
(click-right, st_1_4_400), /*right button calls for task list*/
("N"|"n", st_1_3_300),
("12"|"Twelve"|"twelve"|"TWELVE", st_1_3_12),
("2"|"3"|"4"|"5"|"6"|"9"|"10"|"11"|"13"|"14"|"15", st_1_3_125),
("1"|"One"|"one"|"ONE", st_1_3_100),
("7"|"Seven"|"seven"|"SEVEN", st_1_3_7),
("8"|"Eight"|"eight"|"EIGHT", st_1_3_8),
st_1_3_125))

(st_1_3_7,
(0, clear),
st_1_2_60)

(st_1_3_8,
(0, clear),
st_1_2_70)

(st_1_3_12,
(0, clear),
(assert, shown), st_1_3_45)

(st_1_3_23,
((0, clear),

(0, write, " TITLE: ITEM.
The interface is created by developing the windows the interface user sees and specifying what happens when the user selects an object on the screen or 'pushs' a button. In TAE+ the windows are called panels. There are a number of types of objects that may be positioned on the panel; these objects are called items. Some of the types of items TAE+ provides include:
Data Driven Objects (DDOs) - ex. Stretcher, Strip Chart
Selection Objects - ex. Push button, Checkbox, Icon, Pull down menu
Text Objects - ex. Multi-line field, Label, Keyin, Text display "),
(6, write, "Type 'C' when ready to continue"),
(9, input, keyboard)),
((15 seconds & (past st_1_3_23 wait), (assert, shown), st_1_3_23_1),
(15 seconds, (assert, wait), st_1_3_23),
(click-continue, (assert, shown), st_1_3_23_1),
("C"l"c", (assert, shown), st_1_3_23_1)))

(st_1_3_23_1,
((8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
(9, input, mouse&key)),
(((past st_1_4_26 active), (assert, shown), st_1_4_26),
((past st_1_4_27 active), (assert, shown), st_1_4_27),
((past st_1_4_28_1 active), (assert, shown), st_1_4_28_1),
((past st_1_4_29 active), (assert, shown), st_1_4_29),
((past st_1_4_30 active), (assert, shown), st_1_4_30),
((past st_1_4_31 active), (assert, shown), st_1_4_31),
((past st_1_4_32 active), (assert, shown), st_1_4_32),
((past st_1_4_33 active), (assert, shown), st_1_4_33),
((past st_1_4_34 active), (assert, shown), st_1_4_34),
((past st_1_4_35 active), (assert, shown), st_1_4_35),
((past st_1_4_36 active), (assert, shown), st_1_4_36),
((past st_1_4_37 active), (assert, shown), st_1_4_37),
((past st_1_4_38 active), (assert, shown), st_1_4_38),
((past st_1_4_39 active), (assert, shown), st_1_4_39),
((past st_1_4_40 active), (assert, shown), st_1_4_40),
((past st_1_4_41 active), (assert, shown), st_1_4_41),
(click-left, (assert, shown), st_1_2_1),   /*left button calls for sub-w list*/
(click-middle, (assert, shown), st_1_3_1), /*middle button calls for function list*/
(click-right, (assert, shown), st_1_4_1))) /*right button calls for task list*/

```
(st_1_3_25,
 ((0, clear),
  (0, write, "  TITLE:  ITEM - Creating.
  In order to create new items you must first select the panel upon which the new item will be placed.  Then there are several ways to create items.
First, there is a 'New Item' button on the main TAE+ WorkBench menu.  When this is selected a subsequent screen prompts the user for details
necessary to define the new item.  However, if you have already defined a similar item, you can select this 'similar' item and then Edit/Duplicate on
the TAE+ WorkBench.  An identical item will be generated by TAE+ and 'ghosted' on the panel adjacent to the item it was cloned from.  The only
additional effort necessary is to then select Edit/Modify to replace the TAE+ generated default values, Item Name etc., on the Item Specification
panel."),
  (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
  (9, input, mouse&key)),
 ((click-left, (assert, shown), st_1_2_1),   /*left button calls for sub-w list*/
  (click-middle, (assert, shown), st_1_3_1),  /*middle button calls for function list*/
  (click-right, (assert, shown), st_1_4_1)))  /*right button calls for task list*/

(st_1_3_27,
 ((0, clear),
  (0, write, "  TITLE:  Data Driven Objects (DDOs) - page 1 of 3.
  These provide the ability to graphically model dynamic data as it changes.  Generally, there is a background component and a foreground component
of the display.  The background provides a static picture and the foreground a dynmically driven element that represents the current value of the driving
variable.  For example:  one of the TAE+ provided DDOs is called a stretcher.  A stretcher object is often used to graphically depict a thermometer.
In this case, the background is the static picture of the thermometer and the measured degree markings and the foreground is the dynamic picture
depicting the current value of the thermometer variable.  Further details on how to incorporate DDOs into an interface are provided on subsequent
pages. "),
  (6, write, "Type 'N' for next DDO page."),
  (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respective"),
  (9, input, mouse&key)),
 ((click-left, (assert, shown), st_1_2_1),   /*left button calls for sub-w list*/
  (click-middle, (assert, shown), st_1_3_1),  /*middle button calls for function list*/
  (click-right, (assert, shown), st_1_4_1),   /*right button calls for task list*/
  (click-continue, (assert, shown), st_1_3_27_1),
  ("C"|"c", (assert, shown), st_1_3_27_1),
  ("N"|"n", (assert, shown), st_1_3_27_1)))

(st_1_3_27_1,
```

```
((0, clear),
 (0, write, "  TITLE:  Data Driven Objects (DDOs) - page 2 of 3.
 Lets assume you want to incorporate a thermometer stretcher in your application.  'Select' the panel on which the thermometer DDO will be placed
and click File/Include.  A subwindow will appear querying the user for the name of the file to be included.  Type '$TAEDEMO/res/sun4/ddodemo.res'
in the response area and return. (Do not type the quote marks, just the name.)  A new window will open that contains all of the TAE+ provided DDOs.
'Select' the thermometer object depicted in this window and, holding the mouse-button down, 'drag' it into the desired panel location.  This is now an
item of your interface which can be modified by selecting it, and clicking Edit/Modify."),
 (6, write, "Type 'N' for next DDO page or 'P' for previous DDO page."),
 (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respective"),
 (9, input, mouse&key)),
((click-left, (assert, shown), st_1_2_1),   /*left button calls for sub-w list*/
 (click-middle, (assert, shown), st_1_3_1),  /*middle button calls for function list*/
 (click-right, (assert, shown), st_1_4_1),  /*right button calls for task list*/
 (click-continue, (assert, shown), st_1_3_27_2),
 ("C"|"c", (assert, shown), st_1_3_27_2),
 ("N"|"n", (assert, shown), st_1_3_27_2),
 ("P"|"p", (assert, shown), st_1_3_27)))

(st_1_3_27_2,
 ((0, clear),
  (0, write, "  TITLE:  Data Driven Objects (DDOs) - page 3 of 3.
  In this thermometer example, the interface developer would likely access the Item 'Set Details' and 'Set Contraints' to customize the thermometer high/
low/range settings etc.
  Additional points:
  1.  The TAE+ manual recommends the interface panel to which you are adding the DDO to, be located to the left of the ddodemo.res window.  This
facilitates the user 'dragging' the new object from right-to-left.
  2.  To remove the ddodemo.res file from view:  DO NOT KILL THE WINDOW.  This can corrupt the file.  Select the ddodemo.res file window, by
clicking on it, and then hit the keyboard Delete button."),
  (6, write, "Type 'P' for previous DDO page."),
  (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respective"),
  (9, input, mouse&key)),
 ((click-left, (assert, shown), st_1_2_1),   /*left button calls for sub-w list*/
  (click-middle, (assert, shown), st_1_3_1),  /*middle button calls for function list*/
  (click-right, (assert, shown), st_1_4_1),  /*right button calls for task list*/
  (click-continue, (assert, shown), st_1_3_27),
  ("C"|"c", (assert, shown), st_1_3_27),
```

```
  ("P"|"p", (assert, shown), st_1_3_27)))

(st_1_3_35,
  ((0, clear),
   (0, write, "  ...'presentation cat/type' help text goes here... "),
   (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
   (9, input, mouse&key)),
  ((click-left, (assert, shown), st_1_2_1),    /*left button calls for sub-w list*/
   (click-middle, (assert, shown), st_1_3_1),  /*middle button calls for function list*/
   (click-right, (assert, shown), st_1_4_1)))  /*right button calls for task list*/

(st_1_3_43,
  ((0, clear),
   (0, write, "  TITLE:  PANEL
  The interface is created by developing the windows the interface user sees and specifying what happens when the user selects an object on the screen
or 'pushs' a button.  In TAE+ the windows are called panels.  There are a number of types of objects that may be positioned on the panel; these objects
are called items."),
   (6, write, "Type 'C' when ready to continue"),
   (9, input, keyboard)),
  ((5 seconds & (past st_1_3_43 wait), (assert, shown), st_1_3_43_1),
   (10 seconds, (assert, wait), st_1_3_43),
   (click-continue, (assert, shown), st_1_3_43_1),
   ("C"|"c", (assert, shown), st_1_3_43_1)))

(st_1_3_43_1,
  ((8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
   (9, input, mouse&key)),
  (((past st_1_4_13 active), (assert, shown), st_1_4_13),
   ((past st_1_4_14 active), (assert, shown), st_1_4_14),
   ((past st_1_4_15 active), (assert, shown), st_1_4_15),
   ((past st_1_4_16 active), (assert, shown), st_1_4_16),
   ((past st_1_4_17 active), (assert, shown), st_1_4_17),
   ((past st_1_4_18 active), (assert, shown), st_1_4_18),
   ((past st_1_4_19 active), (assert, shown), st_1_4_19),
   ((past st_1_4_20 active), (assert, shown), st_1_4_20),
   ((past st_1_4_21 active), (assert, shown), st_1_4_21),
```

```
  ((past st_1_4_22 active), (assert, shown), st_1_4_22),
  ((past st_1_4_23 active), (assert, shown), st_1_4_23),
  ((past st_1_4_24 active), (assert, shown), st_1_4_24),
  (click-left, (assert, shown), st_1_2_1),    /*left button calls for sub-w list*/
  (click-middle, (assert, shown), st_1_3_1),  /*middle button calls for function list*/
  (click-right, (assert, shown), st_1_4_1))) /*right button calls for task list*/

(st_1_3_44,
 ((0, clear),
  (0, write, "  TITLE:  PANEL - Creating.
  There are several ways to create panels. First, there is a 'New Panel' button on the main TAE+ WorkBench menu. When this is selected a subsequent
screen prompts the user for details necessary to define the new panel characteristics. However, if you have already defined a similar panel, you can
select this 'similar' item and then select Edit/Duplicate on the TAE+ WorkBench. An identical item will be generated by TAE+ and 'ghosted' on the
screen adjacent to the 'similar' panel it was cloned from. The only additional effort necessary is to then select Edit/Modify to replace the TAE+
generated default values, Panel Name etc., in the Panel Specification window."),
  (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
  (9, input, mouse&key)),
 ((click-left, (assert, shown), st_1_2_1),    /*left button calls for sub-w list*/
  (click-middle, (assert, shown), st_1_3_1),  /*middle button calls for function list*/
  (click-right, (assert, shown), st_1_4_1))) /*right button calls for task list*/

(st_1_3_45,  /* Results when user picks Inter-panel connection in Function-oriented menu.*/
        /* If no Mode change, falls through to show help text, else changes Mode first*/
 ((0, clear), (0, retract, st_1_3_45, active),
  (0, write, "Has the TAE+ WorkBench Mode changed?  (Please enter Y or N)"),
  (9, input, keyboard)),
 ((15 seconds & (past st_1_3_45 wait), (assert, active), st_1_3_250),
  (15 seconds, (assert, wait), st_1_3_45),
  ("YES"|"Yes"|"yes"|"Y"|"y", (assert, active), st_1_1_7),
  ("NO"|"No"|"no"|"N"|"n", st_1_3_45_1),
  (assert, active), st_1_3_250))

(st_1_3_45_1,
 ((0, clear), (0, retract, st_1_3_45, active),
  (0, write, "  ...'inter-panel connections' help text goes here..."),
  (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
```

```
  (9, input, mouse&key)),
  ((click-left, (assert, shown), st_1_2_1),   /*left button calls for sub-w list*/
   (click-middle, (assert, shown), st_1_3_1), /*middle button calls for function list*/
   (click-right, (assert, shown), st_1_4_1))) /*right button calls for task list*/
(st_1_3_52,
  ((0, clear),
   (0, write, "  ...'changing panel dimensions' help text goes here..."),
   (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
   (9, input, mouse&key)),
  ((click-left, (assert, shown), st_1_2_1),   /*left button calls for sub-w list*/
   (click-middle, (assert, shown), st_1_3_1), /*middle button calls for function list*/
   (click-right, (assert, shown), st_1_4_1))) /*right button calls for task list*/

(st_1_3_53,
  ((0, clear),
   (0, write, "  ...'changing panel origin' help text goes here..."),
   (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
   (9, input, mouse&key)),
  ((click-left, (assert, shown), st_1_2_1),   /*left button calls for sub-w list*/
   (click-middle, (assert, shown), st_1_3_1), /*middle button calls for function list*/
   (click-right, (assert, shown), st_1_4_1))) /*right button calls for task list*/

(st_1_3_57,
  ((0, clear),
   (0, write, "  ...'specifying panel titlebar' help text goes here..."),
   (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
   (9, input, mouse&key)),
  ((click-left, (assert, shown), st_1_2_1),   /*left button calls for sub-w list*/
   (click-middle, (assert, shown), st_1_3_1), /*middle button calls for function list*/
   (click-right, (assert, shown), st_1_4_1))) /*right button calls for task list*/

(st_1_3_60,
  ((0, clear),
   (0, write, "  TITLE:  FILE.
   TAE+ stores the interface under development in a file called a 'Resource File'.  The user provides the desired name to store to.  If not provided by the
user, TAE+ appends a .res extension onto the filename."),
```

```
  (6, write, "Type 'C' when ready to continue"),
  (9, input, keyboard)),
 ((15 seconds & (past st_1_3_60 wait), (assert, shown), st_1_3_60_1),
  (15 seconds, (assert, wait), st_1_3_60),
  (click-continue, (assert, shown), st_1_3_60_1),
  ("C"|"c", (assert, shown), st_1_3_60_1)))

(st_1_3_60_1,
 ((8, write,  "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
  (9, input, mouse&key)),
 (((past st_1_4_4 active), (assert, shown), st_1_4_4),
  ((past st_1_4_5 active), (assert, shown), st_1_4_5),
  ((past st_1_4_6 active), (assert, shown), st_1_4_6),
  ((past st_1_4_7 active), (assert, shown), st_1_4_7),
  ((past st_1_4_8 active), (assert, shown), st_1_4_8),
  ((past st_1_4_9 active), (assert, shown), st_1_4_9),
  ((past st_1_4_10 active), (assert, shown), st_1_4_10),
  ((past st_1_4_11 active), (assert, shown), st_1_4_11),
  ((past st_1_4_60 active), (assert, shown), st_1_4_60),
  (click-left, (assert, shown), st_1_2_1),    /*left button calls for sub-w list*/
  (click-middle, (assert, shown), st_1_3_1),  /*middle button calls for function list*/
  (click-right, (assert, shown), st_1_4_1)))  /*right button calls for task list*/

(st_1_3_61,
 ((0, clear),
  (0, write, "  ...'creating new file' help text goes here..."),
  (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
  (9, input, mouse&key)),
 ((click-left, (assert, shown), st_1_2_1),    /*left button calls for sub-w list*/
  (click-middle, (assert, shown), st_1_3_1),  /*middle button calls for function list*/
  (click-right, (assert, shown), st_1_4_1)))  /*right button calls for task list*/

(st_1_3_63,
 ((0, clear),
  (0, write, "  ...'opening existing file' help text goes here... "),
  (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
```

```
 (9, input, mouse&key)),
 ((click-left, (assert, shown), st_1_2_1),   /*left button calls for sub-w list*/
  (click-middle, (assert, shown), st_1_3_1), /*middle button calls for function list*/
  (click-right, (assert, shown), st_1_4_1))) /*right button calls for task list*/

(st_1_3_73,
 ((0, clear),
  (0, write, "  ...'interface startup mode' help text goes here... "),
  (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
  (9, input, mouse&key)),
 ((click-left, (assert, shown), st_1_2_1),   /*left button calls for sub-w list*/
  (click-middle, (assert, shown), st_1_3_1), /*middle button calls for function list*/
  (click-right, (assert, shown), st_1_4_1))) /*right button calls for task list*/

(st_1_3_100,
 ((0, write, "  TITLE:  Function-oriented Topic List.
 This menu or topic list provides the user with a list of help topics ordered alphabetically by function name.  It offers a 'finer grained' listing to facilitate
very specific help topic selection."),
  (6, write, "Type 'C' when ready to continue.  You will be automatically returned to the function-oriented topic list."),
  (9, input, keyboard)),
 ((click-continue, (assert, shown), st_1_3_1),
  ("C"|"c", (assert, shown), st_1_3_1)))

(st_1_3_125,
 ((0, write, "  Selected option not yet implemented."),
  (6, write, "Returning you to your originating menu page..."),
  (0, pause, 5)),
 st_1_3_1)

(st_1_3_150,
 ((0, write, "  Selected option not yet implemented."),
  (6, write, "Returning you to your originating menu page..."),
  (0, pause, 5)),
 st_1_3_300)

(st_1_3_175,
```

```
 ((0, write, "  Selected option not yet implemented."),
  (6, write, "Returning you to your originating menu page..."),
  (0, pause, 5)),
 st_1_3_350)

(st_1_3_200,
 ((0, write, "  Selected option not yet implemented."),
  (6, write, "Returning you to your originating menu page..."),
  (0, pause, 5)),
 st_1_3_400)

(st_1_3_225,
 ((0, write, "  Selected option not yet implemented."),
  (6, write, "Returning you to your originating menu page..."),
  (0, pause, 5)),
 st_1_3_450)

(st_1_3_250,
 ((0, write, "  You have not replied or have entered an invalid answer.  It is assumed that the WorkBench mode has changed."),
  (0, pause, 5)),
 st_1_1_7)

(st_1_3_300,
 ((0, clear),
  (0, write, "   16. Exiting from a resource file    23. Item
                 17.  with save                      24.  Aligning multiple items
                 18.  without save                   25.  Creating
                 19. Exiting from TAE+               26.  Data constraints/null value
                 20. Generating source code          27.  Data Driven Objects (DDOs)
                 21.  Single/multi file generation   28.  Default value
                 22. Including resource file w/i     29.  Deleting item
                     another resource file           30.  Duplicating item(s)

  (Enter topic number, 'N' for next function-oriented page or 'P' for previous function-oriented page)"),
  (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
  (9, input, mouse&key)),
```

```
 ((click-left, st_1_2_150),   /*left button calls for sub-w list*/
  (click-middle, st_1_3_1), /*middle button calls for function list*/
  (click-right, st_1_4_300),  /*right button calls for task list*/
  ("N"|"n", st_1_3_350),
  ("P"|"p", st_1_3_1),
  ("23"|"Twentythree"|"twentythree"|"TWENTYTHREE", st_1_3_23),
  ("25"|"Twentyfive"|"twentyfive"|"TWENTYFIVE", st_1_3_25),
  ("27"|"Twentyseven"|"twentyseven"|"TWENTYSEVEN", st_1_3_27),
  ("17"|"18"|"19"|"20"|"21"|"22"|"24"|"26"|"28"|"29"|"30", st_1_3_150),
  st_1_3_150))

(st_1_3_350,
  ((0, clear),
  (0, write, "  31.  Editing/modifying existing item(s)
                32.  Item attributes                    39. Merging files
                33.    Changing item dim                  40. Modifying multiple items or panels
                34.    Changing item origin               41. Modifying TAE+ WorkBench settings
                35.    Presentation cat/type              42. Okaying a command/action
                36.      Selection Objects
                37.      Specifying data type
                38.      Text Objects

  (Enter topic number, 'N' for next function-oriented page or 'P' for previous function-oriented page)"),
   (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
   (9, input, mouse&key)),
  ((click-left, st_1_2_150),   /*left button calls for sub-w list*/
   (click-middle, st_1_3_1),   /*middle button calls for function list*/
   (click-right, st_1_4_300),    /*right button calls for task list*/
   ("N"|"n", st_1_3_400),
   ("P"|"p", st_1_3_300),
   ("35"|"Thirtyfive"|"thirtyfive"|"THIRTYFIVE", st_1_3_35),
   ("32"|"33"|"34"|"36"|"37"|"38"|"39"|"40"|"41"|"42", st_1_3_175),
   st_1_3_175))

 (st_1_3_400,
   ((0, clear),
```

```
(0, write, "  43. Panel                          51. Panel Attributes
              44.  Creating                     52.  Changing panel dimensions
              45.  Defining inter-panel connects  53.  Changing panel origin
               46.  Deleting                    54.  Set frame chars
               47.  Editing/modifying existing   55.  Set preferred panel state
               48.  Help file                   56.  Set titlebar functionality
               49.  Icon                        57.  Specifying panel titlebar
               50.  Initial/Startup panel       58.  Visibility
                                                59. Quitting TAE+
 (Enter topic number, 'N' for next function-oriented page or 'P' for previous function-oriented page)"),
 (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
 (9, input, mouse&key)),
((click-left, st_1_2_150),   /*left button calls for sub-w list*/
 (click-middle, st_1_3_1),   /*middle button calls for function list*/
 (click-right, st_1_4_1),    /*right button calls for task list*/
 ("N"|"n", st_1_3_450),
 ("P"|"p", st_1_3_350),
 ("43"|"Fourtythree"|"fourtythree"|"FOURTYTHREE", st_1_3_43),
 ("44"|"Fourtyfour"|"fourtyfour"|"FOURTYFOUR", st_1_3_44),
 ("45"|"Fourtyfive"|"fourtyfive"|"FOURTYFIVE", st_1_3_45),
 ("52"|"Fiftytwo"|"fiftytwo"|"FIFTYTWO", st_1_3_52),
 ("53"|"Fiftythree"|"fiftythree"|"FIFTYTHREE", st_1_3_53),
 ("46"|"47"|"48"|"49"|"50"|"51"|"54"|"55"|"56"|"57"|"58"|"59", st_1_3_200),
 st_1_3_200))

(st_1_3_450,
 ((0, clear),
 (0, write, "  60. Resource File        Saving - continued topic
              61.  Creating             68.  to new file
              62.  Editing              69.  part of interface
              63.  Opening existing      70. Sizing
              64.  Merging (Include)     71.  Panel
              65.  Exiting from          72.  Item
              66. Saving                73. Setting the interface startup mode
              67.  to existing file     74. Using Apply, Cancel, Close or OK buttons
```

```
 (Enter topic number or 'P' for function-oriented previous page)"),
 (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
 (9, input, mouse&key)),
((click-left, st_1_2_1),   /*left button calls for sub-w list*/
 (click-middle, st_1_3_1), /*middle button calls for function list*/
 (click-right, st_1_4_1),  /*right button calls for task list*/
 ("P"|"p", st_1_3_400),
 ("60"|"Sixty"|"sixty"|"SIXTY", st_1_3_60),
 ("61"|"Sixtyone"|"sixtyone"|"SIXTYONE", st_1_3_61),
 ("63"|"Sixtythree"|"sixtythree"|"SIXTYTHREE", st_1_3_63),
 ("73"|"Seventythree"|"seventythree"|"SEVENTYTHREE", st_1_3_73),
 ("62"|"64"|"65"|"66"|"67"|"68"|"69"|"70"|"71"|"72"|"74",  st_1_3_225),
 st_1_3_225))


/* 1_4_38 --> 1_4_46 upgrade Novice to Experienced */
/* 1_4_47 --> 1_4_49 upgrade Experienced to Knowledgeable */
/* 1_4_3  --> 1_4_11 downgrade Knowledgeable to Experienced */

(st_1_4_1,
 ((0,clear),
  (0, write, "   1. Task-oriented Topic List      SAVE FILE - continued topic
              2. Overview of TAE+              10.  Saving to an existing file
              3. Starting TAE+                 11.  Doing a partial save
              4. OPEN FILE                     12. WHAT IS A PANEL?
              5.  Opening a new file           13. CREATING A PANEL
              6.  Opening an existing file      14.  Setting panel titlebar
              7.  Merging two files            15.  Defining the panel attributes
              8. SAVE FILE                     16.    Setting panel border
              9.  Saving to a new file          17.    Setting panel visibility/mode
  (Enter topic number or 'N' for next topic-oriented page)"),
  (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
  (9, input, mouse&key)),
 ((click-left, st_1_2_1),   /*left button calls for sub-w list*/
  (click-middle, st_1_3_400), /*middle button calls for function list*/
  (click-right, st_1_4_1),  /*right button calls for task list*/
```

```
  ("N"|"n", st_1_4_300),
  ("4"|"7"|"8"|"9"|"10"|"11"|"15"|"16"|"17", st_1_4_175),
  ("12"|"Twelve"|"twelve"|"TWELVE", st_1_4_12),
  ("13"|"Thirteen"|"thirteen"|"THIRTEEN", st_1_4_13),
  ("14"|"Fourteen"|"fourteen"|"FOURTEEN", st_1_4_14),
  ("1"|"One"|"one"|"ONE", st_1_4_100),
  ("2"|"Two"|"two"|"TWO", st_1_4_125),
  ("3"|"Three"|"three"|"THREE", st_1_4_3),
  ("5"|"Five"|"five"|"FIVE", st_1_4_5),
  ("6"|"Six"|"six"|"SIX", st_1_4_6),
  st_1_4_175))

  /* from 1_4_3 to 1_4_11  if Knowledgeable, downgrade to Experienced */

(st_1_4_3,
 ((0, clear),
  (0, write, "Select topic number:
  1. Resource File
  2. Startup Mode
  3. Other (...results in subsequent menu...)

  (Enter 1, 2 or 3)"),
  (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
  (9, input, mouse&key)),
 ((click-left, (assert, shown), st_1_2_1),    /*left button calls for sub-w list*/
  (click-middle, (assert, shown), st_1_3_1),  /*middle button calls for function list*/
  (click-right, (assert, shown), st_1_4_1),   /*right button calls for task list*/
  ((("1"|"One"|"one"|"ONE") & (past st_1_1_300 knowl_user)), (assert, activeone), st_1_1_200),
  ((("1"|"One"|"one"|"ONE") & (past st_1_1_200 exper_user)), (assert, shown), st_1_3_60),
  (((past st_1_1_200 exper_user) & (past st_1_4_3 activeone)), (assert, shown), st_1_3_60),
  ((("2"|"Two"|"two"|"TWO") & (past st_1_1_300 knowl_user)), (assert, activetwo), st_1_1_200),
  ((("2"|"Two"|"two"|"TWO") & (past st_1_1_200 exper_user)), (assert, shown), st_1_3_73),
  (((past st_1_1_200 exper_user) & (past st_1_4_3 activetwo)), (assert, shown), st_1_3_73),
  ("3"|"Three"|"three"|"THREE", st_1_4_1)))  /*Assumes non-context related*/

(st_1_4_5,
```

```
((0, clear), (0, retract, st_1_4_5, active), (6, write, "please click the mouse ...")),
(((past st_1_1_300 knowl_user), (assert, active), st_1_1_200), /*Knowledgeable shouldn't be asking this question*/
((past st_1_1_200 exper_user), (assert, shown), st_1_3_61),   /*Non-novice sees specific info directly*/
(((past st_1_1_100 novice_user) & (past st_1_3_60 shown)),    /*Novice user shown genl info, then returned to*/
  (assert, shown), st_1_3_61),                                /*show specific info OR novice saw on previous req*/
((past st_1_1_100 novice_user), (assert, active), st_1_3_60)))    /*Novice needs to see genl info*/

(st_1_4_6,
 ((0, clear),
 (0, write, "Select topic number:
 1. Open Resource File
 2. Startup Mode
 3. Other (...results in subsequent menu...)

 (Enter 1, 2 or 3)"),
 (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
 (9, input, mouse&key)),
((click-left, (assert, shown), st_1_2_1),    /*left button calls for sub-w list*/
 (click-middle, (assert, shown), st_1_3_1),  /*middle button calls for function list*/
 (click-right, (assert, shown), st_1_4_1),   /*right button calls for task list*/
 ((("1"|"One"|"one"|"ONE") & (past st_1_1_300 knowl_user)), st_1_4_60),
 ((("1"|"One"|"one"|"ONE") & (past st_1_1_200 exper_user)), (assert, shown), st_1_4_60),
 (((past st_1_1_200 exper_user) & (past st_1_4_6 activeone)), (assert, shown), st_1_4_60),
 ((("2"|"Two"|"two"|"TWO") & (past st_1_1_300 knowl_user)), st_1_4_60),
 ((("2"|"Two"|"two"|"TWO") & (past st_1_1_200 exper_user)), (assert, shown), st_1_3_73),
 (((past st_1_1_200 exper_user) & (past st_1_4_6 activetwo)), (assert, shown), st_1_3_73),
 ("3"|"Three"|"three"|"THREE", st_1_4_1)))  /*Assumes non-context related*/

(st_1_4_12,
  (0, clear),
  (assert, shown), st_1_3_43)

(st_1_4_13,
  ((0,clear), (0, retract, st_1_4_13, active), (6, write, "please click the mouse ...")),
  (((past st_1_1_200 exper_user | past st_1_1_300 knowl_user), (assert, shown), st_1_3_44),
                                  /*Non-novice sees specific info directly*/
```

```
  (((past st_1_1_100 novice_user) & (past st_1_3_43 shown)),(assert, shown), st_1_3_44),
                                    /*Novice user shown genl info, then specific */
  ((past st_1_1_100 novice_user), (assert, active), st_1_3_43)))    /*Novice needs to see genl info*/

(st_1_4_14,
  ((0, clear), (0, retract, st_1_4_14, active), (6, write, "please click the mouse ...")),
  (((past st_1_1_200 exper_user | past st_1_1_300 knowl_user), (assert, shown), st_1_3_57),
                                    /*Non-novice sees specific info directly*/
  (((past st_1_1_100 novice_user) & (past st_1_3_43 shown)),(assert, shown), st_1_3_57),
                                    /*Novice user shown genl info, then specific */
  ((past st_1_1_100 novice_user), (assert, active), st_1_3_43)))  /*Novice needs to see genl info*/

(st_1_4_21,
  ((0, clear), (0, retract, st_1_4_21, active), (6, write, "please click the mouse ...")),
  (((past st_1_1_200 exper_user | past st_1_1_300 knowl_user), (assert, shown), st_1_3_52),
                                    /*Non-novice sees specific info directly*/
  (((past st_1_1_100 novice_user) & (past st_1_3_43 shown)),(assert, shown), st_1_3_52),
                                    /*Novice user shown genl info, then specific */
  ((past st_1_1_100 novice_user), (assert, active), st_1_3_43)))  /*Novice needs to see genl info*/

(st_1_4_22,
  ((0, clear), (0, retract, st_1_4_22, active), (6, write, "please click the mouse ...")),
  (((past st_1_1_200 exper_user | past st_1_1_300 knowl_user), (assert, shown), st_1_3_53),
                                    /*Non-novice sees specific info directly*/
  (((past st_1_1_100 novice_user) & (past st_1_3_43 shown)),(assert, shown), st_1_3_53),
                                    /*Novice user shown genl info, then specific */
  ((past st_1_1_100 novice_user), (assert, active), st_1_3_43))) /*Novice needs to see genl info*/

(st_1_4_25,
  ((0, clear), (0, retract, st_1_4_25, active), (6, write, "please click the mouse ...")),
  (((past st_1_1_200 exper_user | past st_1_1_300 knowl_user), (assert, shown), st_1_3_23),
   ((past st_1_1_100 novice_user), (assert, active), st_1_1_200)))

(st_1_4_26,
  ((0, clear), (0, retract, st_1_4_26, active), (6, write, "please click the mouse ...")),
  (((past st_1_1_200 exper_user | past st_1_1_300 knowl_user), (assert, shown), st_1_3_25),
```

```
                                    /*Non-novice sees specific info directly*/
  (((past st_1_1_100 novice_user) & (past st_1_3_23 shown)), (assert, active), st_1_1_25),
                                    /*Novice user shown genl info, then specific */
  ((past st_1_1_100 novice_user), (assert, active), st_1_3_23)))  /*Novice needs to see genl info*/

(st_1_4_27,
  ((0, clear), (0, retract, st_1_4_27, active), (6, write, "please click the mouse ...")),
  (((past st_1_1_200 exper_user | past st_1_1_300 knowl_user), (assert, shown), st_1_3_35),
                                    /*Non-novice sees specific info directly*/
  (((past st_1_1_100 novice_user) & (past st_1_3_23 shown)),(assert, active), st_1_1_35),
                                    /*Novice user shown genl info, then specific */
  ((past st_1_1_100 novice_user), (assert, active), st_1_3_23)))  /*Novice needs to see genl info*/

(st_1_4_28,
  ((0, clear),
  (0, write," Have you used other kinds of TAE+ objects such as Selection Objects or Text Objects?  (Enter Y or N)"),
   (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
   (9, input, mouse&key)),
  ((click-left, st_1_2_1),    /*left button calls for sub-w list*/
   (click-middle, st_1_3_1),  /*middle button calls for function list*/
   (click-right, st_1_4_1),   /*right button calls for task list*/
   ("Y"|"y"|"Yes"|"yes"|"YES", (assert, obj_user), st_1_4_28_2),
   ("N"|"n"|"No"|"no"|"NO", st_1_4_28_1)))

(st_1_4_28_1,   /*user answered NO to query re used other types of TAE+ objects*/
  ((0, clear), (0, retract, st_1_4_28_1, active), (6, write, "please click the mouse ...")),
  (((past st_1_1_200 exper_user | past st_1_1_300 knowl_user), (assert, shown), st_1_3_27),
                                    /*Non-novice sees specific info directly*/
   (((past st_1_1_100 novice_user) & (past st_1_3_23 shown)),(assert, active), st_1_3_27),
                                    /*Novice user shown genl info, then specific */
   ((past st_1_1_100 novice_user), (assert, active), st_1_3_23)))  /*Novice needs to see genl info*/

(st_1_4_28_2,    /* user answered YES to query re used other types of TAE+ objects*/
  ((0, clear),
   (0, write, "  ...DDOs ala Selection/Text objects...
   For more information see the following additional topics:
```

```
    27.  Selecting a presentation category
    32.  Selecting a presentation type
    33.    DDO -->  Discrete picture, Mover, Rotator
                Stretcher, Strip Chart "),
  (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
  (9, input, mouse&key)),
 ((click-left, st_1_2_1),   /*left button calls for sub-w list*/
  (click-middle, st_1_3_1), /*middle button calls for function list*/
  (click-right, st_1_4_1)))  /*right button calls for task list*/

/* st_1_4_38 to st_1_4_46  if Novice, upgrade to Experienced */

(st_1_4_38,
 ((0, clear), (0, retract, st_1_4_38, active), (6, write, "please click the mouse ...")),
 (((past st_1_1_200 exper_user| past st_1_1_300 knowl_user), (assert, shown), st_1_3_34),
                                  /*Non-novice sees specific info directly*/
  (((past st_1_1_100 novice_user) & (past st_1_3_23 shown)),(assert, active), st_1_1_200),
                                  /*Novice user shown genl info, then upgraded to Exper user */
  ((past st_1_1_100 novice_user), (assert, active), st_1_3_23)))  /*Novice needs to see genl info*/

(st_1_4_39,
 ((0, clear), (0, retract, st_1_4_39, active), (6, write, "please click the mouse ...")),
 (((past st_1_1_200 exper_user | past st_1_1_300 knowl_user), (assert, shown), st_1_3_33),
                                  /*Non-novice sees specific info directly*/
  (((past st_1_1_100 novice_user) & (past st_1_3_23 shown)),(assert, active), st_1_1_200),
                                  /*Novice user shown genl info, then upgraded to Exper user */
  ((past st_1_1_100 novice_user), (assert, active), st_1_3_23)))  /*Novice needs to see genl info*/

/* st_1_4_47 to st_1_4_49  if Experienced, upgrade to Knowledgeable */

(st_1_4_44,
 (0, clear),
 (assert, shown), st_1_3_45)

(st_1_4_60,  /* originates from 1_4_6 */
 ((0, clear), (0, retract, st_1_4_60, active), (6, write, "please click the mouse ...")),
```

```
(((past st_1_1_300 knowl_user), (assert, active), st_1_1_200), /*Knowledgeable shouldn't be asking this*/
 ((past st_1_1_200 exper_user), (assert, shown), st_1_3_63),   /*Exper sees specific info directly*/
 (((past st_1_1_100 novice_user) & (past st_1_3_60 shown)),    /*Novice user shown genl info, then returned to */
  (assert, shown), st_1_3_63),                                 /*show specific info OR novice saw on previous req*/
 ((past st_1_1_100 novice_user), (assert, active), st_1_3_60)))  /*Novice needs to see genl info*/



(st_1_4_100,
 ((0, write, "  TITLE:  Task-oriented Topic List.
 This menu or topic list provides the user with a list of help topics ordered based on the TAE+ tasks frequently utilized in creating an interface.  The
ordering is meant to assist the user in determining 'what's next' in the development process, as well as providing a structure for organizing the help
topics."),
  (6, write, "Type 'C' when ready to continue.  You will automatically be returned to the task-oriented topic list."),
  (9, input, keyboard)),
 ((click-continue, (assert, shown), st_1_4_1),
  ("C"|"c", (assert, shown), st_1_4_1)))

(st_1_4_125,
 ((0, write, "  TITLE:  Overview of TAE+.
 Transportable Applications Environment Plus (TAE+) is a portable software development environment that supports rapid building, tailoring and
management of graphic-oriented user interfaces.  It uses MIT's XWindow System (Ver 11, Rel 4) and the Open Software Foundation's (OSF's) Motif
Toolkit.  It can generate code in C, Fortran and Ada as well as high level TAE Command Language (TCL)."),
  (6, write, "Type 'C' when ready to continue.  You will automatically be returned to the task-oriented topic list."),
  (9, input, keyboard)),
 ((click-continue, (assert, shown), st_1_4_1),
  ("C"|"c", (assert, shown), st_1_4_1)))

(st_1_4_175,
 ((0, write, "  Selected option not yet implemented."),
  (6, write, "Returning you to your originating menu page..."),
  (0, pause, 5)),
 st_1_4_1)

(st_1_4_200,
```

```
 ((0, write, "  Selected option not yet implemented."),
  (6, write, "Returning you to your originating menu page..."),
  (0, pause, 5)),
 st_1_4_300)

(st_1_4_225,
 ((0, write, "  Selected option not yet implemented."),
  (6, write, "Returning you to your originating menu page..."),
  (0, pause, 5)),
 st_1_4_350)

(st_1_4_250,
 ((0, write, "  Selected option not yet implemented."),
  (6, write, "Returning you to your originating menu page..."),
  (0, pause, 5)),
 st_1_4_400)

(st_1_4_300,
 ((0,clear),
  (0, write, "   19.  Adding a related help file      25. WHAT IS AN ITEM?
                20.  Adding a related panel icon    26. CREATING AN ITEM
                21.  Positioning the panel          27.  Selecting a presentation category
                22.  Sizing the panel               28.     Data Driven Objects (DDOs)
                23.  Specifying first/opening panel 29.     Selection Objects
                24.  Specifying a default panel     30.     Text Objects
                                                    31.     X-Workspace

  (Enter topic number, 'N' for next topic-oriented page or 'P' for previous topic-oriented page)"),
  (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
  (9, input, mouse&key)),
 ((click-left, st_1_2_150),   /*left button calls for sub-w list*/
  (click-middle, st_1_3_300),   /*middle button calls for function list*/
  (click-right, st_1_4_1),    /*right button calls for task list*/
  ("N"|"n", st_1_4_350),
  ("P"|"p", st_1_4_1),
  ("21"|"Twentyone"|"twentyone"|"TWENTYONE", st_1_4_21),
```

```
 ("22"|"Twentytwo"|"twentytwo"|"TWENTYTWO", st_1_4_22),
 ("25"|"Twentyfive"|"twentyfive"|"TWENTYFIVE", st_1_4_25),
 ("26"|"Twentysix"|"twentysix"|"TWENTYSIX", st_1_4_26),
 ("27"|"Twentyseven"|"twentyseven"|"TWENTYSEVEN", st_1_4_27),
 ("28"|"Twentyeight"|"twentyeight"|"TWENTYEIGHT", st_1_4_28),
 ("19"|"20"|"23"|"24"|"29"|"30"|"31", st_1_4_200),
 st_1_4_200))


(st_1_4_350,
 ((0, clear),
  (0, write, "  CREATING AN ITEM - continued topic
  32.  Selecting a presentation type
  33.     DDO -->  Discrete picture, Mover, Rotator, Stretcher, Strip Chart
  34.     Selection Object --> Push button, Checkbox, Icon, Pull down
                        menu, Radio button,  Scale, Selection list
  35.     Text Object --> Multi-line field, Label, Keyin, Text display
  36.     X-Workspace --> x-workspace
  37.  Constraining the data entry --> Strings, integers, real numbers
 (Enter topic number, 'N' for next topic-oriented page or 'P' for previous topic-oriented page)"),
  (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
  (9, input, mouse&key)),
 ((click-left, st_1_2_150),    /*left button calls for sub-w list*/
  (click-middle, st_1_3_350),  /*middle button calls for function list*/
  (click-right, st_1_4_1),     /*right button calls for task list*/
  ("N"|"n", st_1_4_400),
  ("P"|"p", st_1_4_300),
  ("32"|"33"|"34"|"35"|"36"|"37", st_1_4_225),
  st_1_4_225))

(st_1_4_400,
 ((0,clear),
  (0, write, "38.  Positioning the item on the panel
             39.  Sizing the item                  44. LINKING PANELS WITH PANEL CONNECTIONS
             40.  Defining item appearance         45. REHEARSING THE INTERFACE
             41.  Specifying item default value    46. GENERATING SOURCE CODE
```

```
   42.  Specifying item details          47. FINISHING THE INTERFACE
   43. USING ALIGNMENT                  48.  Coding the event/event stubs
                                        49. QUITTING TAE+

(Enter topic number or 'P' for previous topic-oriented page)"),
  (8, write, "Left-Middle-Right mouse buttons provide subwindow, function- and task-oriented topic lists respectively."),
  (9, input, mouse&key)),
 ((click-left, st_1_2_150),    /*left button calls for sub-w list*/
  (click-middle, st_1_3_350),  /*middle button calls for function list*/
  (click-right, st_1_4_1),     /*right button calls for task list*/
  ("P"|"p", st_1_4_350),
  ("44"|"Fourtyfour"|"fourtyfour"|"FOURTYFOUR", st_1_4_44),
  ("38"|"39"|"40"|"41"|"42"|"43"|"45"|"46"|"47"|"48"|"49", st_1_4_250),
  st_1_4_250))
```

APPENDIX F

CFD GRAPH GRAMMAR (as modified Jan 93)

(Extracted from [MASKE 92])

```
cfd_graph ::= cfd_graph cfd_node | cfd_node | cfd_menu cfd_node
cfd_node ::= "(" cfd_id "," action_list "," response_list ")" | identifier ":=" string
cfd_menu ::= "(" "menu" string menulist ")"
menulist ::= string "->" cfd_id "," menulist | string "->" cfd_id
cfd_id ::= identifier
action_list ::= "(" act_node_list ")" | action_node
response_list ::= "(" res_node_list ")" | response_node
action_node ::= "(" region_id "," action ")"
act_node_list ::= act_node_list "," action_node | action_node
response_node ::= "(" pattern "," cfd_id ")"
  | "(" pattern "," "(" "assert" "," identifier ")" "," cfd_id ")"
  | "(" pattern "," "ignore" ")"
  | "(" pattern "," "(" "assert" "," identifier ")" "," "ignore" ")"
  | "(" "assert" "," identifier ")" "," cfd_id | cfd_id
res_node_list ::= res_node_list "," response_node | response_node
region_id ::= integer | region_id "+" integer
action ::= "draw" "," identfier
  | "draw" "," identifier "@" location
  | "clear" | "clear" "," identifier "@" location
  | "write" "," string | "input" "," input_list
  | "pause" "," integer | "drag" "," identifier
  | "quit" | "retract" "," cfd_id "," identifier
input_list ::= "mouse" | "keyboard" | "mouse&key"
location ::= "(" loc_part "," loc_part ")"
loc_part ::= loc_part "+" term | loc_part "-" term | term
term ::= term "*" factor | term "/" factor | factor
factor ::= integer | "mouseX" | "mouseY"
  | identifier ".x" | identifier ".y"
  | "halfwid" | "halfht"
  | "(" loc_part ")"
patpart ::= keywords | "click-left" | "click-right" | "click-middle"
  | "click-any" | loc_part relop loc_part | "click-exit" | "click-help"
  | "click-continue" | "mouse-move" | integer "seconds"
  | "past" cfd_id identifier | "(" pattern ")"
patconj ::= patpart | patconj "&" patpart
pattern ::= patconj | pattern "|" patconj
relop ::= "==" | ">" | "<" | ">=" | "<="
keywords ::= string
```

REFERENCES

[BOREN 85] Borenstein, N. S., *The Design and Evaluation of Online Help Systems*, Ph.D. Dissertation, Carnegie-Mellon University, (location??),1985.

[BURTO 82] Burton, R. R. and Brown, J. S., "An Investigation of Computer Coaching for Informal Learning Activities", *Intelligent Tutoring Systems*, Sleeman, D. and Brown, J. S. eds., Academic Press, 1982.

[CHERR 89] Cherry, J. M. and Jackson, S., "Online Help: Effects of Content and Writing Style on User Performance and Attitudes", *IEEE Transactions on Professional Communication*, v. 32, n. 4, pp. 294-299, December 1989.

[DUFFY 85] Duffy, T. M. and Langston, M. D., "Online Help Design Issues for Authoring Systems", November 1985.

[FARRA 92] Farrand, A. B. and Wolfe, S. J., "On-line Help: Are We Tossing the Users a Lifesaver or an Anchor?", *CHI'92 Posters and Short Talks*, Monterey, California, May 1992.

[GWEI 90] Gwei, G. M. and Foxley, E., "Towards a Consultative On-line Help System", *International Journal of Man-Machine Studies*, v. 32, n. 4, pp. 363-383, April 1990.

[HOUGH 90] Houghton, R. C., "Online Help Systems: A Conspectus", *Communications of the ACM*, v. 27, n. 2, pp. 126-133, February 1984.

[JACKS 92] Jackson, S., Cherry, J., and Fryer, B., "Online Scenario-Based Task Help", *IEEE Transactions on Professional Communication*, v. 35, n. 2, pp. 91-97, June 1992.

[MAASS 83] Maase, S., "Why Systems Transparency?", *Psychology of Computer Use*, pp. 19-27, Academic Press, 1983.

[MASKE 92] Maskell, D. M., *Concept-Flow Diagrams: Method for Design of Computer-Aided Instruction*, Master's Thesis, Naval Postgraduate School, Monterey, California, March 1992.

[MOILY 87] Moily, J. P., Murray, T. J. and Agarwal, R., *A Preliminary Specification of an On-Line Expert Help System*, Information & Management, n. 13, pp. 191-196, 1987.

[NASA 91] NASA Goddard Space Flight Center, *TAE Plus User Interface Developer's Guide*, V. 5.1, p. 8, April 1991.

[RELLE 81] Relles, N. and Sondheimer, N. K., "A Unified Approach to Online Assistance", pp. 383-388, AFIPS Conference Proceedings, 1981.

[RIDGW 87] Ridgway, Lenore, S., "Read My Mind: What Users Want from Online Information", *IEEE Transactions on Professional Communication*, v. 30, n. 2, pp. 87-90, June 1987.

[SELLE 90] Sellen, A. and Nicol, A., "Building User-Centered On-line Help", *The Art of Human-Computer Interface Design*, pp. 143-153, Laurel, B. ed., Addison Wesley Publishing Company, Inc., 1990.

[YOURD 89] Yourdon, E., *Modern Structural Analysis*, pp. 259-265, Yourdon Press, 1989.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexanderia, VA 22304-6145	2
2.	Dudley Knox Library Code 52 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	CDR Gary J. Hughes, SC, USN Acting Chairman Computer Science Department Naval Postgraduate School Monterey, CA 93943-5000	1
4.	Dr. Timothy J. Shimeall Code CS/Sm Assistant Professor, Computer Science Department Naval Postgraduate School Monterey, CA 93943-5000	3
5.	Dr. David A. Erickson Code CS/Er Adjunct Professor, Computer Science Department Naval Postgraduate School Monterey, CA 93943-5000	3
6.	Commander Space and Naval Warfare Systems Command PMW 183-11A ATTN: LT D. M. Maskell Washington, DC 20363-5100	1
7.	LCDR.John A. Daley, USN Code CS/Da Naval Postgraduate School Monterey, CA 93943-5000	1
8.	LT Nancy J. McClellan c/o 408 Pine Park Court Martinez, CA 94553	2